This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Learning Bottleneck Concepts in Image Classification

Bowen Wang[1], Liangzhi Li[2]*, Yuta Nakashima[2], Hajime Nagahara[2]
Osaka University, Japan
[1]bowen.wang@is.ids.osaka-u.ac.jp
[2]{li, n-yuta, nagahara}@ids.osaka-u.ac.jp

Abstract

Interpreting and explaining the behavior of deep neural networks is critical for many tasks. Explainable AI provides a way to address this challenge, mostly by providing per-pixel relevance to the decision. Yet, interpreting such explanations may require expert knowledge. Some recent attempts toward interpretability adopt a concept-based framework, giving a higher-level relationship between some concepts and model decisions. This paper proposes Bottleneck Concept Learner (BotCL), which represents an image solely by the presence/absence of concepts learned through training over the target task without explicit supervision over the concepts. It uses self-supervision and tailored regularizers so that learned concepts can be human-understandable. Using some image classification tasks as our testbed, we demonstrate BotCL's potential to rebuild neural networks for better interpretability [1].

1. Introduction

Understanding the behavior of deep neural networks (DNNs) is a major challenge in the explainable AI (XAI) community, especially for medical applications [19, 38], for identifying biases in DNNs [2, 18, 42], *etc*. Tremendous research efforts have been devoted to the post-hoc paradigm for *a posteriori* explanation [29, 33]. This paradigm produces a relevance map to spot regions in the input image that interact with the model's decision. Yet the relevance map only tells *low-level* (or per-pixel) relationships and does not explicitly convey any semantics behind the decision. Interpretation of relevance maps may require expert knowledge.

The *concept-based* framework [22, 37, 50] is inspired by the human capacity to learn a new concept by (subconsciously) finding finer-grained concepts and reuse them in different ways for better recognition [24]. Instead of giving per-pixel relevance, this framework offers higher-level relationships between the image and decision mediated by a limited number of *concepts*. That is, the decision is explained by giving a set of concepts found in the image. The interpretation of the decision is thus straightforward once the interpretation of each concept is established.

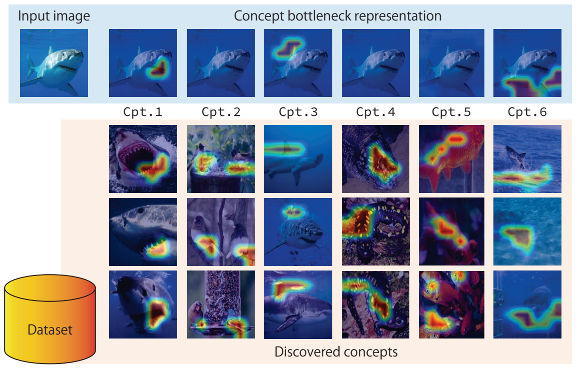


Figure 1. Examples of concepts discovered by BotCL in ImageNet [10] and concepts in the input image. BotCL automatically discovers a set of concepts optimized for the target task and represents an image solely with the presence/absence of concepts.

Some works use concepts for the post-hoc paradigm for better interpretation of the decision [14, 50], while the link between the decision and concepts in the image is not obvious. The *concept bottleneck* structure [23] uses the presence/absence of concepts as image representation (referred to as *concept activation*). The classifier has access only to the concept activation, so the decision is strongly tied to the concepts. This bottleneck structure has become the mainstream of the concept-based framework [5, 20, 28, 31].

A major difficulty in this framework is designing a set of concepts that suits the target task. A promising approach is handcrafting them [4, 21, 48], which inherently offers better interpretability at the cost of extra annotations on the concepts. Recent attempts automatically discover concepts [1, 13, 14, 46]. Such concepts may not always be consistent with how humans (or models) see the world [25, 47] and may require some effort to interpret them, but concept discovery without supervision is a significant advantage.

*Corresponding author.
[1]Code is avaliable at https://github.com/wbw520/BotCL and a simple demo is available at https://botcl.liangzhili.com/.

Inspired by these works, we propose **bottleneck concept learner** (BotCL) for simultaneously discovering concepts and learning the classifier. BotCL optimizes concepts for the given target image classification task without supervision for the concepts. An image is represented solely by the existence of concepts and is classified using them. We adopt a slot attention-based mechanism [26, 27] to spot the region in which each concept is found. This gives an extra signal for interpreting the decision since one can easily see what each learned concept represents by collectively showing training images with the detected concepts. Figure 1 shows examples from ImageNet [10]. BotCL discovers a predefined number of concepts in the dataset, which are exemplified by several images with attention maps. An image of `Great White Shark` is represented by the right part of mouth (`Cpt.1`) and fins (`Cpt.3`). BotCL uses a single fully-connected (FC) layer as a classifier, which is simple but enough to encode the co-occurrence of each concept and each class.

Contribution. For better concept discovery, we propose to use *self-supervision over concepts*, inspired by the recent success in representation learning [9, 16]. Our ablation study demonstrates that self-supervision by contrastive loss is the key. We also try several constraints on concepts themselves, *i.e*., *individual consistency* to make a concept more selective and *mutual distinctiveness* for better coverage of various visual elements. These additional constraints regular the training process and help the model learn concepts of higher quality.

2. Related Works

2.1. Explainable AI

XAI focuses on uncovering black-box deep neural networks [3, 6, 12, 32, 35, 36, 41, 43, 46]. A major approach is generating a relevance map that spots important regions for the model's decision. Various methods have been designed for specific architectures, *e.g*., CAM [49], and Grad-CAM [33] for convolutional neural networks; [7] for Transformers [40]. However, the interpretation of the relevance maps may not always be obvious, which spurs different approaches [34, 45], including context-based ones.

2.2. Concept-based framework for interpretability

A straightforward way to define a set of concepts for a target task is to utilize human knowledge [22,48]. Such concepts allow quantifying their importance for a decision [21]. A large corpus of concepts [4, 39] is beneficial for delving into hidden semantics in DNNs [50]. These methods are of the post-hoc XAI paradigm, but a handcrafted set of concepts can also be used as additional supervision for models with the concept bottleneck structure [15, 22, 31].

Handcrafting a set of concepts offers better interpretability as they suit human perception; however, the annotation cost is non-negligible. Moreover, such handcrafted concepts may not always be useful for DNNs [47]. These problems have motivated automatic concept discovery. Superpixels are a handy unit for finding low-level semantics, and concepts are defined by clustering them [13, 14, 30]. Another interesting approach is designing a set of concepts to be sufficient statistics of original DNN features [46]. These methods are designed purely for interpretation, and concept discovery is made aside from training on the target task.

The concept bottleneck structure allows optimizing a set of concepts for the target task. ProtoPNet [8] adopts this structure and identifies concepts based on the distance between features and concepts. SENN [1] uses self-supervision by reconstruction loss for concept discovery.

SENN inspired us to use self-supervision, but instead of reconstruction loss, we adopt contrastive loss tailored. For a natural image classification task, this contrastive loss is essential for concept discovery.

3. Model

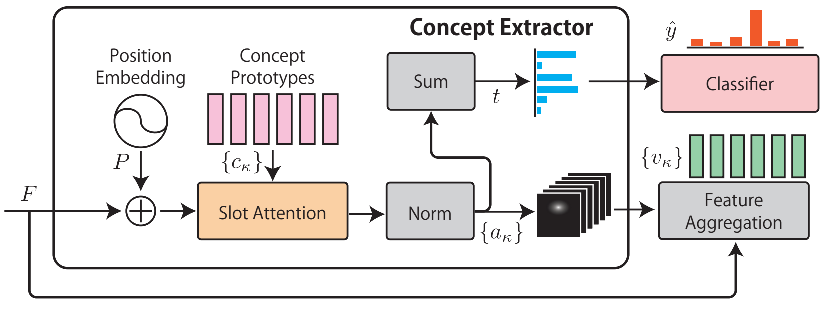
Given a dataset $\mathcal{D} = \{(x_i, y_i)|i = 1, 2, \dots, N\}$, where x_i is an image and y_i is the target class label in the set Ω associated with x_i. BotCL learns a set of k concepts while learning the original classification task. Figure 2a shows an overview of BotCL's training scheme, consisting of a concept extractor, regularizers, and a classifier, as well as self-supervision (contrastive and reconstruction losses).

For a new image x, we extract feature map $F = \Phi(x) \in \mathbb{R}^{d\times h\times w}$ using a backbone convolutional neural network Φ. F is then fed into the concept extractor g_C, where C is a matrix, each of whose κ-th column vector c_κ is a *concept prototype* to be learned. The concept extractor produces concept bottleneck activations $t \in [0, 1]^k$, indicating the presence of each concept, as well as concept features $V \in \mathbb{R}^{d\times k}$ from regions where each concept exists. The concept activations in t are used as input to the classifier to compute score $s \in [0, 1]^{|\Omega|}$. We use self-supervision and regularizers for training, taking t and V as input to constrain the concept prototypes.

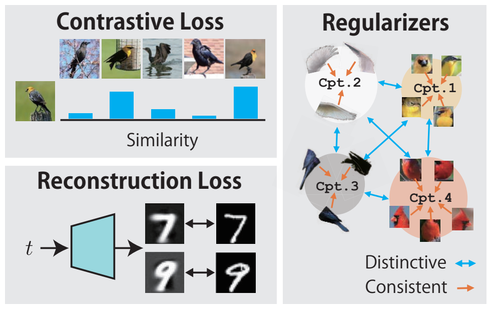
3.1. Concept Extractor

Concept extractor uses slot attention [26, 27]-based mechanism to discover visual concepts in $\mathcal{D}$. We first add position embedding P to feature map F to retain the spatial information, *i.e*., $F' = F + P$. The spatial dimension of F' is flattened, so its shape is $l \times d$, where $l = hw$.

The slot-attention computes attention over the spatial dimension for concept κ from c_κ and F'. Let $Q(c_\kappa) \in \mathbb{R}^d$, and $K(F') \in \mathbb{R}^{d\times l}$ denote nonlinear transformations for c_κ and F', respectively, given as multi-layer perceptrons with three FC layers and a ReLU nonlinearity between them. Attention $a_\kappa \in [0, 1]^l$ is given using a normalization function



(a)



(b)

Figure 2. (a) The model pipeline. (b) Self-supervision and regularizers.

ϕ (refer to **supp. material**) as

$$a_\kappa = \phi(Q(c_\kappa)^\top K(F')). \tag{1}$$

This attention indicates where concept κ presents in the image as shown in Figure 1. If concept κ is absent, corresponding entries of a_κ are all close to 0. We summarize the presence of each concept into concept activation t_κ by reducing the spatial dimension of a_κ as $t_\kappa = \tanh(\sum_m a_{\kappa m})$, where $a_{\kappa m}$ is the m-th element of a_κ.

3.2. Feature Aggregation

For training, we also aggregate features in F corresponding to concept κ into concept feature v_κ by

$$v_\kappa = F a_\kappa, \tag{2}$$

which gives the average of image features over the spatial dimension weighted by attention.

3.3. Classifier

We use a single FC layer without a bias term as the classifier, and concept activation $t = (t_1, \dots, t_k)^\top$ is the only input, serving as the concept bottleneck [22]. Formally, letting W be a learnable matrix, prediction $\hat{y} \in \mathbb{R}^{|\Omega|}$ is given by

$$\hat{y} = Wt. \tag{3}$$

This classifier can be roughly interpreted as learning the correlation between the class and concepts. Let w_ω be the raw vector of W corresponding to class $\omega \in \Omega$, and $w_{\omega\kappa}$ is its κ-th element. A positive value of $w_{\omega\kappa}$ means that concept κ co-occurs with class ω in the dataset, so its presence in a new image positively supports class ω. Meanwhile, a negative value means the concept rarely co-occurs.

4. Training

4.1. Self-supervision for Concept Discovery

The absence of concept labels motivates us to incorporate self-supervision for concept discovery. We employ two losses for different types of target tasks.

Reconstruction loss. SENN [1] uses an autoencoder-like structure for learning better representation. We assume this structure works well when visual elements are strongly tied with the position[2] since even discrete concepts should have sufficient information to reconstruct the original image. Based on this assumption, we design a reconstruction loss for self-supervision. As shown in Figure 2b, decoder D only takes t as input and reconstructs the original image. We define our reconstruction loss as

$$l_{\text{rec}} = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \|D(t) - x\|^2. \tag{4}$$

Contrastive loss. The composition of natural images is rather arbitrary, so information in t should be insufficient to reconstruct the original image. we thus design a simple loss for an alternative, borrowing the idea from the recent success of contrastive learning for self-supervision [9, 16].

We leverage the image-level labels of the target classification task. Let $\hat{t} = 2t - \mathbf{1}_k$, where $\mathbf{1}_k$ is the k-dimensional vector with all elements being 1. If a pair $(\hat{t}, \hat{t}')$ of concept activations belong to the same class (*i.e.*, $y = y'$ for y and y' corresponding to $\hat{t}$ and $\hat{t}'$), they should be similar to each other since a similar set of concepts should be in the corresponding images, and otherwise dissimilar. The number $|\Omega|$ of classes can be smaller than the number $|\mathcal{B}|$ of images in a mini-batch so that a mini-batch can have multiple images of the same class. Therefore, we use sigmoid instead of softmax, leading to

$$l_{\text{ret}} = -\frac{1}{|\mathcal{B}|} \sum \alpha(y, y') \log J(\hat{t}, \hat{t}', y, y'), \tag{5}$$

where α is the weight to mitigate the class imbalance problem (see **supp. material**) and

$$J(\hat{t}, \hat{t}', y, y') = \begin{cases} \sigma(\hat{t}^\top \hat{t}') & \text{for } y = y' \\ 1 - \sigma(\hat{t}^\top \hat{t}') & \text{otherwise} \end{cases}. \tag{6}$$

[2]For example, images of "7" in MNIST almost always have the acute angle in the top-right part.

4.2. Concept Regularizers

We also employ concept regularizers to facilitate training. They constrain concept prototypes $\{c_\kappa\}$ through $\{v_\kappa\}$.

Individual consistency. For better interpretability, each learned concept should not have large variations. That is, the concept features v_κ and v'_κ of different images should be similar to each other if t_κ is close to 1. Let $\mathcal{H}_\kappa$ denote the set of all concept features of different images in a mini-batch, whose activation is larger than the empirical threshold ξ, which is dynamically calculated as the mean of t_κ in a mini-batch. Using the cosine similarity $\text{sim}(\cdot, \cdot)$, we define the consistency loss as:

$$l_{\text{con}} = -\frac{1}{k} \sum_{\kappa} \sum_{v_\kappa, v'_\kappa} \frac{\text{sim}(v_\kappa, v'_\kappa)}{|\mathcal{H}_\kappa|(|\mathcal{H}_\kappa| - 1)}, \tag{7}$$

where the second summation is computed over all combinations of concept features v_κ and v'_κ. This loss penalizes a smaller similarity between v_κ and v'_κ.

Mutual distinctiveness. To capture different aspects of images, different concepts should cover different visual elements. This means that the average image features of concept κ within a mini-batch, given by $\bar{v}_\kappa = \sum_{v_\kappa \in \mathcal{H}_\kappa} v_\kappa$, should be different from any other $v_{\kappa'}$. We can encode this into a loss term as

$$l_{\text{dis}} = \sum_{\kappa, \kappa'} \frac{\text{sim}(\bar{v}_\kappa, \bar{v}_{\kappa'})}{k(k-1)}, \tag{8}$$

where the summation is computed over all combinations of concepts. Note that concept κ is excluded from this loss if no image in a mini-batch has concept κ.

4.3. Quantization Loss

Concept activation t can be sufficiently represented by a binary value, but we instead use a continuous value for training. We design a quantization loss to guarantee values are close to 0 or 1, given by

$$l_{\text{qua}} = \frac{1}{k|\mathcal{B}|} \sum_{x \in \mathcal{B}} \left\| \text{abs}(\hat{t}) - \mathbf{1}_\kappa \right\|^2, \tag{9}$$

where $\text{abs}(\cdot)$ gives the element-wise absolute value and $\|\cdot\|$ gives the Euclidean norm.

4.4. Total Loss

We use softmax cross-entropy for the target classification task's loss, donated by l_{cls}. The overall loss of BotCL is defined by combining the losses above as

$$L = l_{\text{cls}} + \lambda_{\text{R}} l_{\text{R}} + \lambda_{\text{con}} l_{\text{con}} + \lambda_{\text{dis}} l_{\text{dis}} + \lambda_{\text{qua}} l_{\text{qua}}, \tag{10}$$

where l_{R} is either l_{rec} or l_{ret} depending on the target domain, λ_{qua}, λ_{con}, λ_{dis}, and λ_R are weights to balance each term.

5. Results

5.1. Experimental Settings

We evaluate BotCL on MNIST [11], CUB200 [44], and ImageNet [10]. For evaluating discovered concepts, we re-generated a synthetic shape dataset (Synthetic) [46].

For MNIST, we applied the same networks as [1] for the backbone and the concept decoder. For CUB200 (same data split as [22]) and ImageNet, we used pre-trained ResNet [17] as the backbone with a 1×1 convolutional layer to reduce the channel number (512 for ResNet-18 and 2048 for ResNet-101) to 128. We chose a concept number $k = 20$ for MNIST and $k = 50$ for the other natural image datasets. To generate Synthetic, we followed the setting of [46], where 18,000 images were generated for training and 2,000 for evaluation. We used $k = 15$ with ResNet-18 backbone.

Images were resized to 256×256 and cropped to 224×224 (images in Synthetic were directly resized to 224×224). Only random horizontal flip was applied as data augmentation during training. The weight of each loss was defaulted to $\lambda_{\text{qua}} = 0.1$, $\lambda_{\text{con}} = 0.01$, $\lambda_{\text{dis}} = 0.05$, and $\lambda_{\text{R}} = 0.1$.

5.2. Classification Performance

We compare the performance of BotCL with corresponding baselines (LeNet for MNIST and ResNet-18 for others with a linear classifier), our reimplementation of k-means and PCA in [46],[3] and state-of-the-art concept-based models. Table 1 summarized the results. BotCL with contrastive loss (BotCL$_{\text{Cont}}$) achieves the best accuracy on CUB200, ImageNet, and Synthetic, outperforming the baseline linear classifiers. It is also comparable to the state-of-the-art on MNIST and Synthetic. BotCL with reconstruction loss (BotCL$_{\text{Rec}}$) shows a performance drop over CUB200, ImageNet, and Synthetic, while it outperforms BotCL$_{\text{Cont}}$ on MNIST. This behavior supports our assumption that the reconstruction loss is useful only when concepts are strongly tied to their spatial position. Otherwise, t is insufficient to reconstruct the original image, and BotCL fails. Contrastive self-supervision is the key to facilitating concept discovery.

We also explore the relationship between the number of classes and BotCL's accuracy over CUB200 and ImageNet. We used small and large variants of ResNet as the backbone. We extracted subsets of the datasets consisting of the first n classes along with the class IDs. Figure 3 shows that BotCL has a competitive performance when the number of classes is less than 200. We conclude that BotCL hardly degrades the classification performance on small- or middle-sized datasets. However, this is not the case for $n > 200$ (refer to **supp. material** for larger n and different k's).

[3]Implementation details are in **supp. material.**

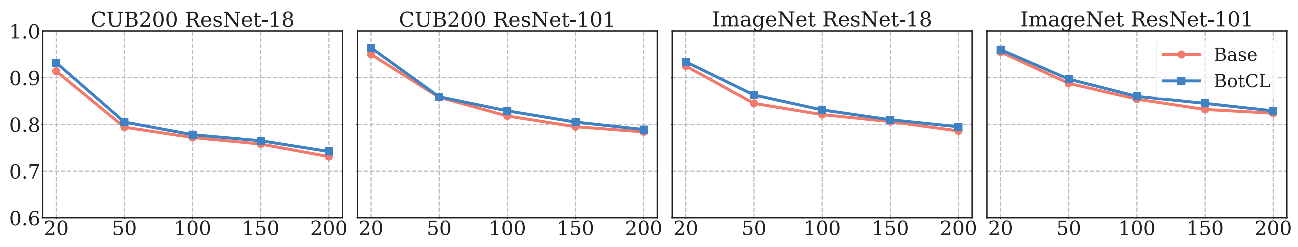


Figure 3. Classification accuracy vs. the number of classes. We used subsets of CUB200 and ImageNet with $k = 50$ and ResNet-18 and ResNet-101 backbones.

Table 1. Performance comparison in classification accuracy. The best concept-based method is highlighted in bold. $\text{BotCL}_{\text{Rec}}$ and $\text{BotCL}_{\text{Cont}}$ are both BotCL but with reconstruction and contrastive loss, respectively. For ImageNet, we used the first 200 classes.

	CUB200	ImageNet	MNIST	Synthetic
Baseline	0.731	0.786	0.988	0.999
k-means* [46]	0.063	0.427	0.781	0.747
PCA* [46]	0.044	0.139	0.653	0.645
SENN [1]	0.642	0.673	**0.985**	0.984
ProtoPNet [8]	0.725	0.752	0.981	0.992
$\text{BotCL}_{\text{Rec}}$	0.693	0.720	0.983	0.785
$\text{BotCL}_{\text{Cont}}$	**0.740**	**0.795**	0.980	**0.998**

5.3. Interpretability

5.3.1 Qualitative validation of discovered concepts

Figure 4a visualizes a_κ, showing concept κ in the image, over MNIST. We selected 5 concepts out of 20 that are most frequently activated (*i.e.*, $t_\kappa > 0.5$) in the training set.[4] Taking digits `0` and `9` as an example, we can observe that they share `Cpts.3-5` and the only difference is `Cpt.2` that locates in the lower edge of the vertical stroke of `9`. This stroke is specific to digit `9`. We used $\text{BotCL}_{\text{Rec}}$, so we can remove `Cpt.2` before reconstruction, which generates an image like 0 (refer to Section 5.3.3). Some concepts are incompatible with human intuition; yet we can interpret such concepts (*e.g.*, `Cpt.1` may attend to the missing stroke that completes a circle).

For CUB200, we train $\text{BotCL}_{\text{Const}}$ with $n = 50$ and $k = 20$. Figure 5a shows the attention maps of an image of `yellow headed black bird`. We can observe that the attentions for `Cpts.1-5` cover different body parts, including the head, wing, back, and feet, which proves that BotCL can learn valid concepts from the natural image as well. **Supp. material** exemplifies all concepts discovered from MNIST and CUB200.

[4]`Cpts.1-5` are ordered based on the frequency counted in the dataset.

5.3.2 Consistency and distinctiveness of each concept

BotCL is designed to discover individually consistent and mutually distinctive concepts. We qualitatively verify this by showing each concept with its top-5 activated images[5] with attention maps in Figure 4b. For MNIST, different concepts cover different patterns, and each concept covers the same patterns in different samples (even the samples of different classes). Figure 5b for CUB200 shows that BotCL renders a similar behavior on the CUB200 dataset; that is, the top-5 concepts are responsible for different patterns, and each of them is consistent.

5.3.3 Contribution of each concept in inference

We can qualitatively see the contribution of each concept by removing the concept and seeing the changes in the corresponding self-supervision task's output. As shown in Figure 4c, when we set the activation of `Cpt.2` (responsible for the vertical stroke of digit `9`) to zero, the reconstructed image looks like digit `0`. When `Cpt.1`, representing the absence of the circle in digit `7`, is deactivated (*i.e.*, t_1 is set to 0), a circle emerges in the upper part of the reconstructed image. The resulting image looks more like digit `9`.

For CUB200 shown in Figure 5c, we show images most similar to the input image in Figure 5a among the dataset in terms of $\hat{t}^\top \hat{t}'$, with ablating each concept. When `Cpt.1` (responsible for the yellow head) is deactivated, more black-head bird images appear in the top-8 images. `Cpt.5` covers birds' feet and is common among most bird classes. Deactivating this concept does not change the top-8 images. These observations suggest that although some concepts do not contribute to classification performance, images are successfully represented by combinations of concepts.

5.4. Quantitative Evaluation on Synthetic

One problem of the concept-based approach is the absence of established quantitative evaluations of concepts because the choice of concepts may be arbitrary and the same level of representability may be achieved with different sets of concepts. A single predefined set of concepts

[5]For each concept κ, five images whose t_κ is highest among $\mathcal{D}$.

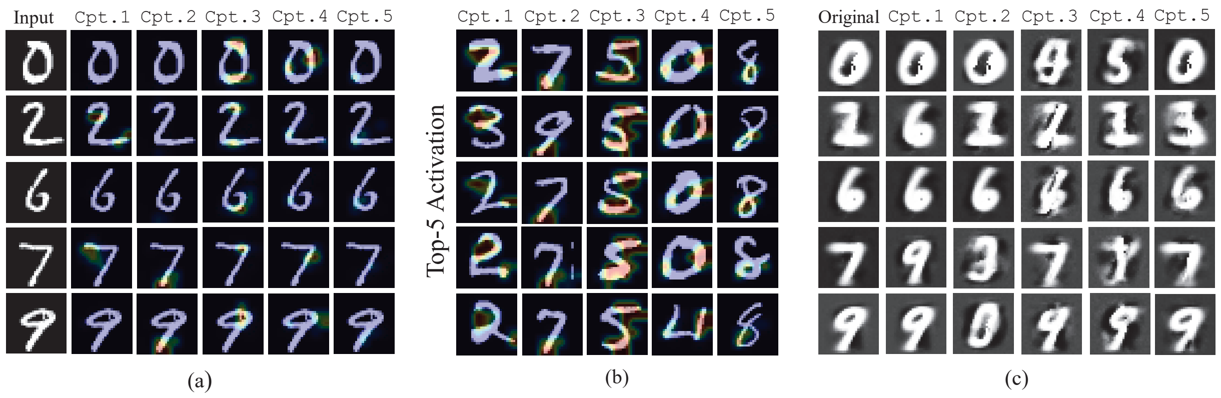


Figure 4. Concepts for MNIST. (a) Attention maps for different input images. (b) Top-5 activated images (images in the dataset whose t_κ is largest) for each concept. (c) Images reconstructed by our concept decoder with all detected concepts (original) and with a certain concept deactivated.

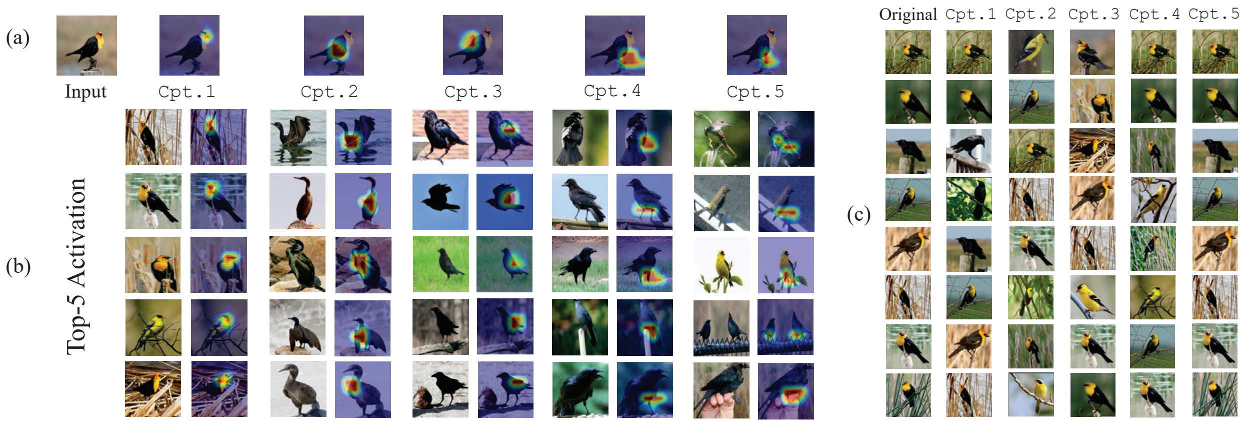


Figure 5. Concepts learned for CUB200. (a) Visualization of 5 most important concepts for `yellow headed black bird`. (b) Top-5 activated concepts. (c) Image retrieval when all detected concepts were used (original) and when a certain concept was deactivated.

is not enough to evaluate the goodness of discovered concepts. Literature has evaluated concepts qualitatively (as Section 5.3) or by user study (as Section 5.5).

We decided to use Synthetic [46] for quantitatively evaluating concepts. The task is a multi-label classification that involves 15 shapes. Combinations of the 5 shapes (shown in Figure 6a, `S.1` to `S.5`) form 15 classes, and the other 10 shapes are noises[6]. We deem a shape is covered by concept κ when the shape's area and concept κ's area (the area with $a_\kappa > \gamma$ for BotCL, where $\gamma = 0.9$ is a predefined threshold) overlap. Let $h_{s\kappa} = 1$ denote shape s overlaps with concept κ, and $h_{s\kappa} = 0$ otherwise. The coverage of s by concept κ is given by

$$\text{Coverage}_{s\kappa} = \mathbb{E}[h_{s\kappa}], \tag{11}$$

where the expectation is computed over the images in the test set with concept κ activated. The concepts and the 5 shapes are associated as a combinatorial optimization problem so that the sum of $\text{Coverage}_{s\kappa}$ over s are maximized.

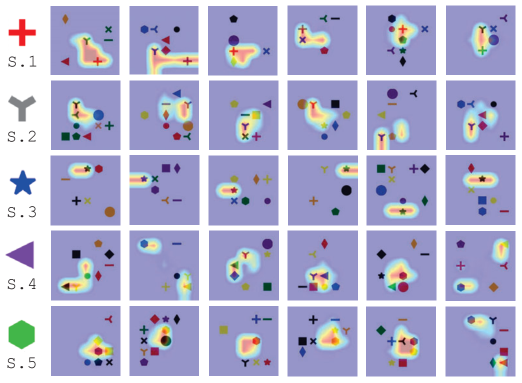
We use $k = 15$ to train BotCL. Figure 6a visualizes the concept associated with each shape[7]. A concept is located by a_κ for 6 images with the highest concept activations t_κ. The concepts cover the associated shapes with relatively small regions, but one concept usually covers multiple shapes. This can be further evident in Figure 6b that shows $\text{Coverage}_{s\kappa}$. `Cpt.8` only covers `S.3`, whereas `Cpt.1` and `Cpt.13` covers multiple shapes.

We use three metrics other than accuracy to evaluate the performance of concept discovery[8]: (i) **Completeness** measures how well a concept covers its associated shape in the dataset. (ii) **Purity** shows the ability to discover concepts that only cover a single shape. (iii) **Distinctiveness** quanti-

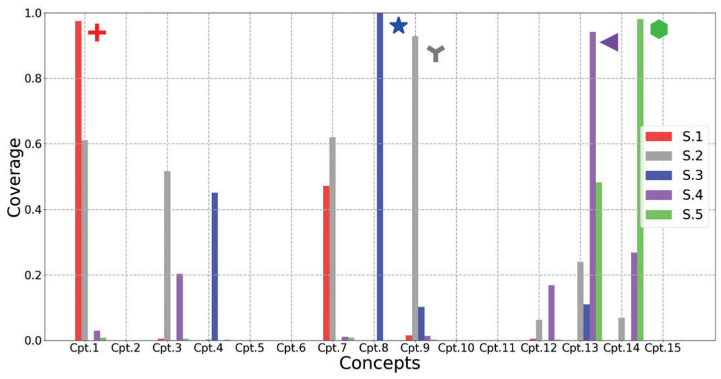
[6]Images are generated with random shapes, so there can be multiple classes (combinations of shapes) in a single image, which forms a multi-label classification task.

[7]Note that in this experiment, only shapes matter but not colors.

[8]Further details are in **supp. material**.



(a)



(b)

Figure 6. Experiment on Synthetic. (a) S.1-S.5 are the five shapes of which combinations form classes. Attention maps next to each shape are of the concept that covers the shape. (b) Coverage$_{s\kappa}$ (the concept associated with each of the five shapes is marked).

fies the difference among concepts based on the coverage.

BotCL with contrastive loss is compared[9] with ACE [14], and two baselines PCA and k-means in [46]. We apply k-means or PCA to F of all images in the dataset after flattening the spatial dimensions. The cluster centers or the principal components are deemed as concepts. Attention maps can be computed by Euclidean distance or cosine similarity. Once the attention maps are obtained, we follow BotCL's process for classification.

As shown in Table 2, BotCL shows better completeness, distinctiveness, and accuracy scores than comparative methods. Although k-means is able to discover concepts, they are not optimized for the target classification task, and the performance is low. As we discussed, the concepts learned by BotCL tend to cover more than one target shape, causing a comparatively low purity. The cluster center of k-means is able to capture only one kind of shape at the cost of completeness. We can also observe that all methods are affected by concept number k, and generally a larger k ensure better performance on all metrics. This result is not surprising, but we confirmed that a larger k is preferable for better interpretability. We detail the generation of the dataset, implementation of PCA and k-means, and formal definitions of metrics in **supp. material**.

5.5. User Study

Our user study is designed to evaluate BotCL with realistic datasets for the challenge of human understanding. Participants are asked to observe the test images with the attention map for concept κ (refer to Section 5.3.2) and select some phrases in the predefined vocabulary that best describes the concept (*i.e.*, attended regions). They can also choose *None of them* if they cannot find any consistent visual elements. We recruited 20 participants for each concept of MNIST and 30 participants for CUB200 using Amazon Mechanical Turk.

[9]SENN [1] and ProtoPNet [8] are not comparable. SENN's concepts globally cover a whole image. ProtoPNet requires way more concepts.

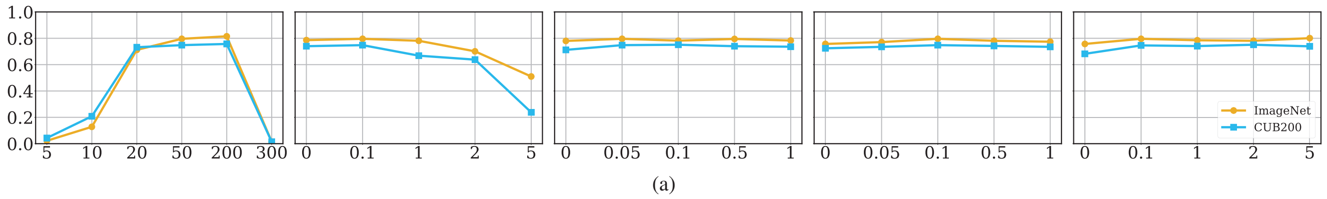
Table 2. Quantitative evaluation on Synthetic. Note that ACE uses concepts for post-hoc explanation and does not use them for classification. Comp., Dist., and Acc. mean completeness, distinctiveness, and accuracy, respectively.

		Comp.	Purity	Dist.	Acc.
$k=5$	ACE	0.662	0.274	0.084	—
	k-means	0.630	0.724	0.215	0.652
	PCA	0.458	0.170	0.298	0.571
	BotCL	0.618	0.453	0.281	0.835
$k=15$	ACE	0.614	0.221	0.151	—
	k-means	0.816	**0.978**	0.272	0.747
	PCA	0.432	0.162	0.286	0.645
	BotCL	**0.925**	0.744	**0.452**	**0.998**

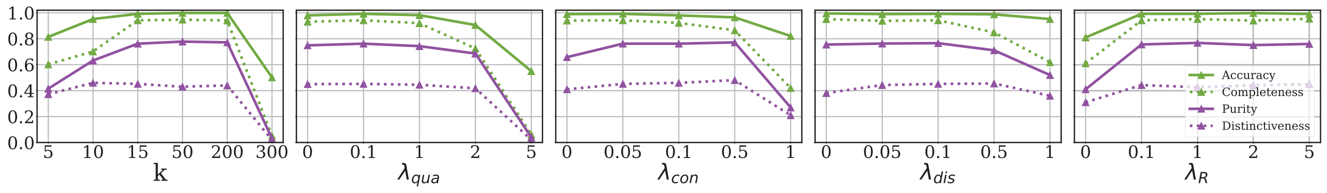
Table 3. Results of our user study.

		CDR ↑		CC ↑		MIC ↓	
Dataset	Concepts	Mean	Std	Mean	Std	Mean	Std
MNIST	Annotated	1.000	0.000	0.838	0.150	0.071	0.047
	BotCL	0.825	0.288	0.581	0.274	0.199	0.072
	Random	0.122	0.070	0.163	0.074	0.438	0.039
CUB200	Annotated	0.949	0.115	0.595	0.113	0.512	0.034
	BotCL	0.874	0.156	0.530	0.116	0.549	0.036
	Random	0.212	0.081	0.198	0.039	0.574	0.031

We defined three metrics to summarize the participants' responses. (i) **Concept discovery rate** (CDR): The ratio of the responses that are not *None of them* to all responses. A higher CDR means participants can find some consistent visual elements for many concepts. (ii) **Concept consistency** (CC): The ratio of exact matches out of all pairs of participants' responses. A high value means many participants use the same phrases to describe a concept. (iii) **Mutual information between concepts** (MIC): The similarity of the response distribution, computed over all possible pairs of concepts. This value is high when multiple concepts cover



(a)



(b)

Figure 7. Results of ablation study. (a) Hyperparameter values vs. classification accuracy on ImageNet and CUB200. (b) Hyperparameter values vs. classification accuracy and other metrics on Synthetic.

the same visual elements; therefore, lower is better.

For comparison, we also evaluated a manual annotation[10] and random scribbling for the same images. Table 3 shows that BotCL yields good scores for all metrics on both datasets (close to the manual annotation), showing the learned concepts are interpretable for humans (from CDR), consistent (from CC), and mutually distinct (from MIC). More details are in **supp. material**.

5.6. Ablation Study

We conducted ablation studies using the default hyperparameters except for the one to be explored. As there is no ground truth concept for CUB200 and ImageNet, only accuracy is evaluated (Figure 7a). For Synthetic, accuracy and the three metrics in Section 5.4 are employed (Figure 7b).

Impact of k. A small k decreases the accuracy and other metrics, which means the necessity of searching the minimum number of concepts. Also, training tends to fail for all datasets when k is large (detailed in the **supp. materials**). The number of concepts should be tuned for each dataset. This sensitivity is one of BotCL's limitations.

Impact of λ_{qua}. This hyperparameter controls how close t should be to a binary. The accuracy and the other metrics worsen when λ_{qua} gradually increases. BotCL encodes some information into t (such as the area that a concept occupies), which is lost for larger λ_{qua}. An extreme value may also cause vanishing gradients.

Impact of λ_{con} and λ_{dis}. The individual consistency and mutual distinctiveness losses hardly affect the performance on CUB200 and ImageNet, although we can see a slight drop when the values are zero for CUB200. For Synthetic, the performance metrics vary as they are designed to be. Meanwhile, the accuracy is relatively insensitive to these hyperparameters. The choice of concepts may be highly arbitrary, and different sets of concepts may achieve similar classification performance. This arbitrariness may allow the designing of dedicated concept regularizers for the target task. However, training failures happen when they are set to be large. A small value benefits training.

[10]The authors annotated.

Impact of λ_R. Due to the lower performance of the reconstruction loss, we studied the impact of the contrastive loss only. The contrastive loss almost always improves the classification accuracy. The performance boost is significant in CUB200 and Synthetic. As ImageNet has more training data, this may imply that self-supervision greatly contributes to the learning of concepts when training samples are limited. These results demonstrate the importance of the contrastive loss. This is interesting since this loss uses the same labels as the classification loss.

6. Conclusion

This paper presents BotCL for learning bottleneck concepts. Our qualitative and quantitative evaluation showed BotCL's ability to learn concepts without explicit supervision on them but through training for a target classification task. We also demonstrated that BotCL could provide interpretability on its decision and learned concepts themselves.
Limitations. One limitation of BotCL is that it requires tuning the number k of concepts for each dataset. It might be an interesting research direction to estimate k, *e.g.*, based on the number of classes in a given classification task. We will investigate the phenomenon to mitigate this problem.

Acknowledgement This work is partly supported by JST CREST Grant No. JPMJCR20D3, JST FOREST Grant No. JPMJFR216O, JSPS KAKENHI Grant-in-Aid for Scientific Research (A). This work is also supported by JSPS KAKENHI Grant Number 19K10662, 20K23343, 21K17764, and 22H03353.

References

[1] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *NeurIPS*, 2018.

[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, and Richard Benjamins. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, pages 6541–6549, 2017.

[5] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Semantic bottleneck for computer vision tasks. In *ACCV*, pages 695–712, 2018.

[6] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *WACV*, pages 839–847, 2018.

[7] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *CVPR*, pages 782–791, 2021.

[8] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. *NeurIPS*, 32, 2019.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[11] Li Deng. The mnist database of handwritten digit images for machine learning research. *Signal Processing Magazine*, 29(6):141–142, 2012.

[12] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*, pages 2950–2958, 2019.

[13] 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 *CVPR*, pages 2195–2204, 2021.

[14] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. Towards automatic concept-based explanations. *NeurIPS*, 2019.

[15] Ju He, Jie-Neng Chen, Shuai Liu, Adam Kortylewski, Cheng Yang, Yutong Bai, and Changhu Wang. Transfg: A transformer architecture for fine-grained recognition. In *AAAI*, pages 852–860, 2022.

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[18] Yusuke Hirota, Yuta Nakashima, and Noa Garcia. Quantifying societal bias amplification in image captioning. In *CVPR*, pages 13450–13459, 2022.

[19] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312, 2019.

[20] Zixuan Huang and Yin Li. Interpretable and accurate fine-grained recognition via region grouping. In *CVPR*, pages 8662–8672, 2020.

[21] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors TCAV. In *ICML*, pages 2668–2677, 2018.

[22] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *ICML*, pages 5338–5348, 2020.

[23] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. Attribute and simile classifiers for face verification. In *ICCV*, pages 365–372, 2009.

[24] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[25] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294*, 2019.

[26] Liangzhi Li, Bowen Wang, Manisha Verma, Yuta Nakashima, Ryo Kawasaki, and Hajime Nagahara. Scouter: Slot attention-based classifier for explainable image recognition. In *ICCV*, pages 1046–1055, 2021.

[27] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *NeurIPS*, 2020.

[28] Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic bottleneck networks. *arXiv preprint arXiv:1907.10882*, 2019.

[29] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. *BMVC*, 2018.

[30] Andres Felipe Posada-Moreno, Nikita Surya, and Sebastian Trimpe. ECLAD: Extracting concepts with local aggregated descriptors. *arXiv preprint arXiv:2206.04531*, 2022.

[31] Mattia Rigotti, Christoph Miksovic, Ioana Giurgiu, Thomas Gschwind, and Paolo Scotton. Attention-based interpretability with concept transformers. In *ICLR*, 2022.

[32] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. *arXiv preprint arXiv:2001.00396*, 2020.

[33] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *CVPR*, pages 618–626, 2017.

[34] Botian Shi, Lei Ji, Pan Lu, Zhendong Niu, and Nan Duan. Knowledge aware semantic concept expansion for image-text matching. In *IJCAI*, volume 1, page 2, 2019.

[35] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, pages 3145–3153, 2017.

[36] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014.

[37] Wolfgang Stammer, Marius Memmel, Patrick Schramowski, and Kristian Kersting. Interactive disentanglement: Learning concepts by interacting with their prototype representations. In *CVPR*, pages 10317–10328, 2022.

[38] Bas HM van der Velden, Hugo J Kuijf, Kenneth GA Gilhuijs, and Max A Viergever. Explainable artificial intelligence (XAI) in deep learning-based medical image analysis. *Medical Image Analysis*, page 102470, 2022.

[39] Saurabh Varshneya, Antoine Ledent, Robert A Vandermeulen, Yunwen Lei, Matthias Enders, Damian Borth, and Marius Kloft. Learning interpretable concept groups in CNNs. *IJCAI*, 2021.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[41] Bowen Wang, Liangzhi Li, Manisha Verma, Yuta Nakashima, Ryo Kawasaki, and Hajime Nagahara. Mtunet: Few-shot image classification with visual explanations. In *CVPR workshops*, pages 2294–2298, 2021.

[42] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. Designing theory-driven user-centric explainable AI. In *Proc. CHI conference on human factors in computing systems*, pages 1–15, 2019.

[43] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-CAM: Score-weighted visual explanations for convolutional neural networks. In *CVPR workshops*, pages 24–25, 2020.

[44] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD birds 200. 2010.

[45] Liang Wang Yan Huang, Qi Wu. Learning semantic concepts and order for image and sentence matching. In *CVPR*, 2018.

[46] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. In *NeurIPS*, volume 33, pages 20554–20565, 2020.

[47] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018.

[48] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene CNNs. *ICLR*, 2015.

[49] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016.

[50] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In *ECCV*, pages 119–134, 2018.

Supplementary Materials of
Learning Bottleneck Concepts in Image Classification

Contents

1. Unveiling Learned Concepts

Figure 1a shows the activations of all 20 concepts for digit `0` to `9`, learned from MNIST [4]. We can see that each digit only has a few concepts activated, *e.g.*, digit `7` has `Cpt.3`, `Cpt.8`, `Cpt.11`, `Cpt.12`, and `Cpt.15`. We also show the top-10 activated samples for each concept (*i.e.*, for concept κ, the samples with the highest ten t_κ's in the training set) in Figure 1b. It can be observed that some concepts are hardly activated. For instance, `Cpt.1` has no significant highlights, suggesting smaller t_κ.

Figure 1c provides the reconstruction results by our concept decoder when a certain concept is deactivated by setting corresponding t_κ being zero (reconstructed images with significant visual changes are marked in red). We can see that digit `7` turns into digit `9` when `Cpt.3` is deactivated. Figure 1d shows that this change happens consistently for all samples of digit `7`.

In addition, as our classifier is a single fully-connected (FC) layer, we can easily obtain the contribution of concept κ to class ω as $I_{\omega\kappa} = t_\kappa z_{\omega\kappa}$, where $z_{\omega\kappa}$ is the (ω, κ)-th element of the learnable matrix Z of the FC layer in Eq. (13) in the main paper. Figure 1e gives the importance of each concept for the digit `7` shown in Figure 1a. We can see that `Cpt.3` and `Cpt.8` are among the most decisive concepts.

Figure 2 shows the attention map for each concept for the input image (the left-most image) and $z_{\omega\kappa}$'s for $\omega =$ `yellow headed black bird`. We see that the classification of `yellow headed black bird` is mainly based on `Cpt.2` and `Cpt.16`, which look to represent the breast and head, respectively. Figure 3 show 10 example images with the attention map for each concept learned from a 50-class subset of CUB200 [12], where the 10 images are of the highest t_κ. We can see that the attended regions of most concepts are consistent among its top-10 activated samples, and most concepts look to represent meaningful patterns. For example, `Cpt.7` focuses on the leg, and `Cpt.8` focuses on wings.

The concepts learned on the ImageNet dataset [3] are shown in Figure 6. We can see that, for the given sample of `Goldfish` (Figure 4a), the two most important concepts are `Cpt.2` and `Cpt.9`. These two concepts cover semantically consistent regions in the training samples (according to Figure 4b) and look to represent dorsal fins and a near-gill region, respectively.

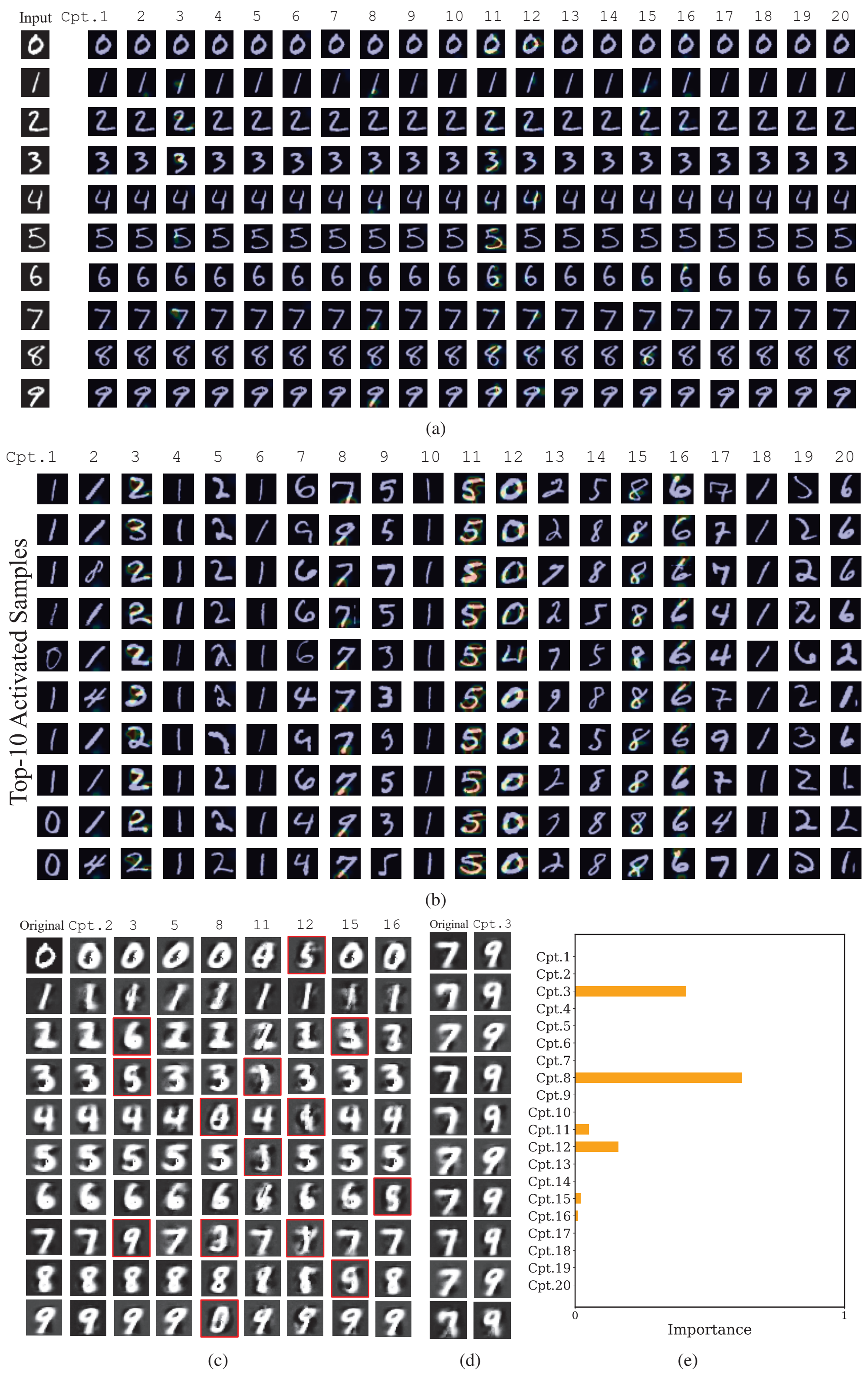


Figure 1. (a) Attention map for each of 20 concepts extracted for the input (left-most) image of each digit. (b) Top-10 activated samples for each concept. (c) Image reconstruction with one concept deactivated. (d) Image reconstruction for different samples of digit `7` with deactivating `Cpt.3`. (e) Concept importance for digit `7`.

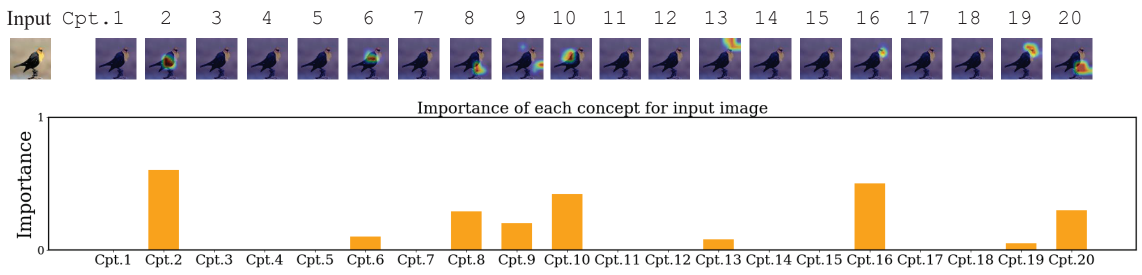


Figure 2. Concept activations for a sample of `yellow headed black bird`.

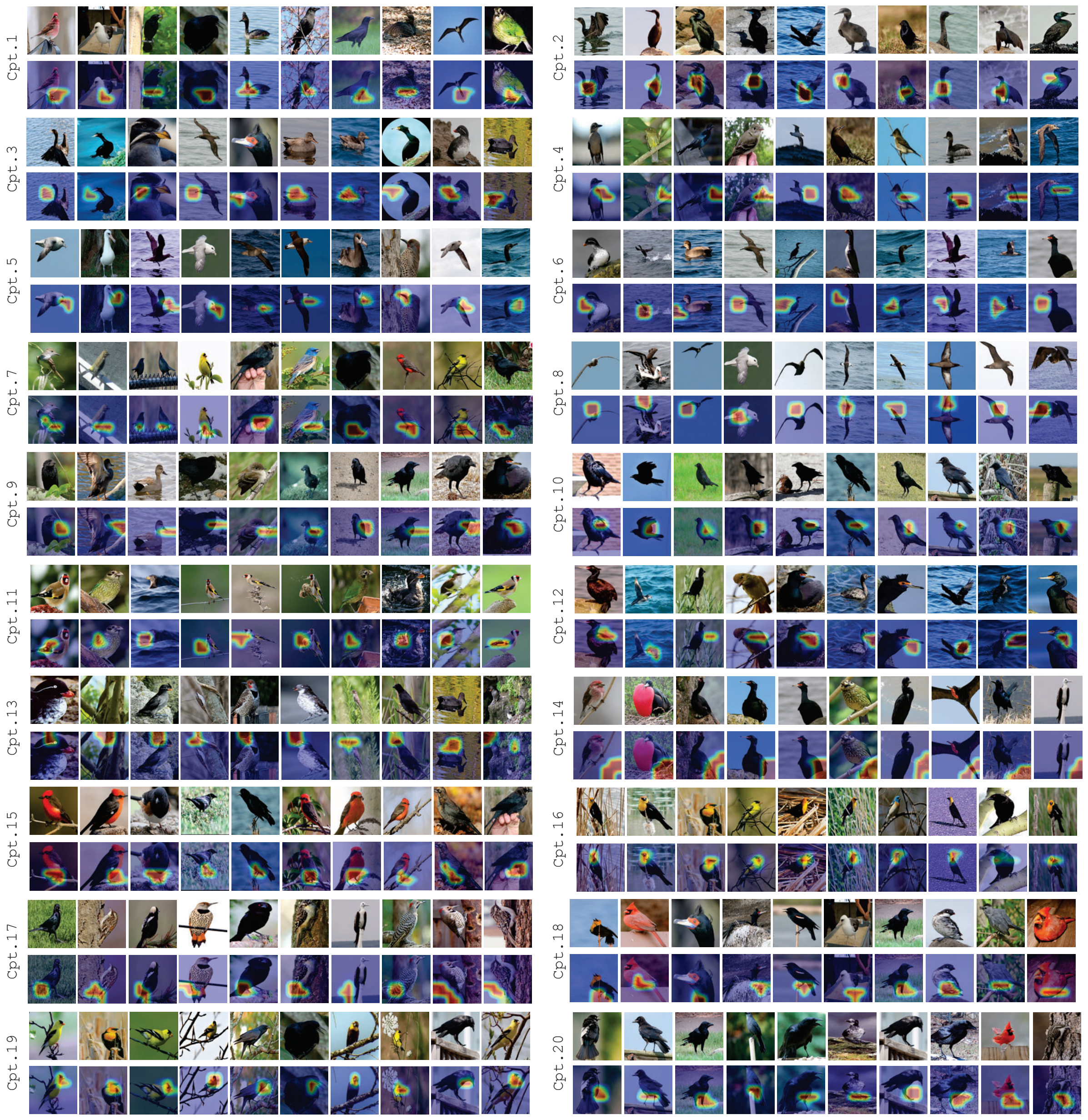
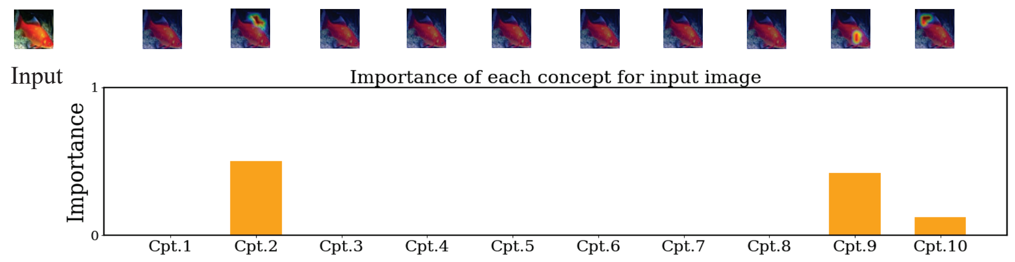


Figure 3. Concepts learned from CUB200, represented by top-10 activated samples.

(a)



(b)

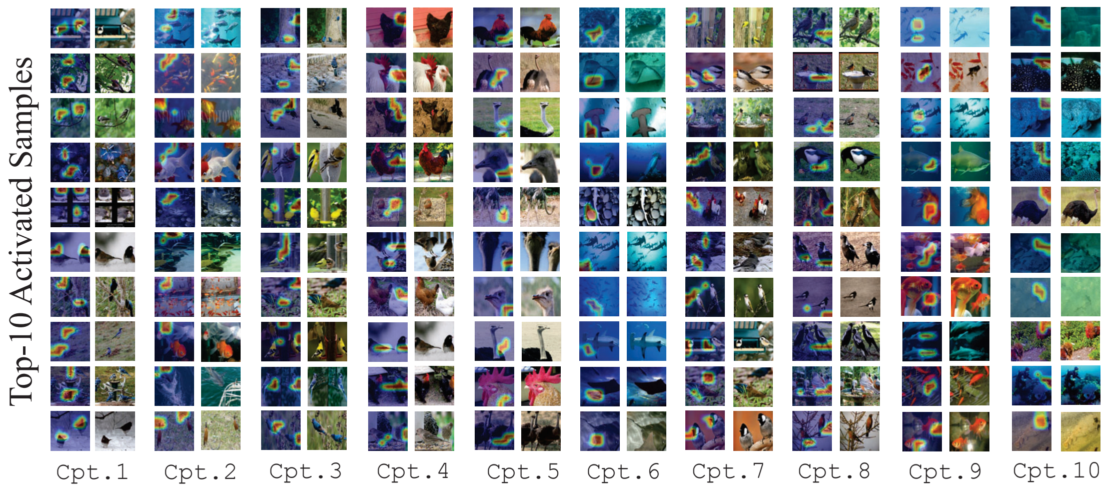


Figure 4. (a) Concept activations for a sample of `Goldfish` and the importance of each concept. (b) Concepts learned from ImageNet ($n = 20$ and $k = 10$).

2. Details of Experiments Settings

2.1. Normalization Function ϕ

The normalization ϕ determines the spatial distribution of each concept, which may depend on the target domain. For example, images for the handwritten digit recognition dataset are typically in black and white, and only the shape formed by strokes matters. In this case, concepts are less likely to overlap with each other spatially. Meanwhile, natural images have colors, textures, and shapes; any (combination) of them can be a concept. Thus, concepts possibly coincide at the same spatial position.

Let $a'_k = Q(c_\kappa)^\top K(F')$ (appears in Eq. (1)). For domain with supposedly non-overlapping concepts (*e.g.*, MNIST), we use ϕ given by

$$\phi_\kappa(\{a'_\kappa\}) = \sigma(a'_\kappa) \odot \text{softmax}_\text{S}(\{a'_\kappa\}). \tag{1}$$

This normalization takes $\{a'_\kappa\}$ for all concepts as input, which slightly abuses Eq. (1) of the main paper. σ is the (element-wise) sigmoid function, and $\odot$ is the Hadamard product. $\text{softmax}_\text{S}(\cdot)$ is taken over all concepts at each spatial position, so different concepts are less likely to be detected at the same spatial position.

For domains with overlapping concepts (*e.g.*, CUB200 and ImageNet), we only use the sigmoid function for normalization as

$$\phi(a'_\kappa) = \sigma(a'_\kappa). \tag{2}$$

2.2. Weight $\alpha(y, y')$

Equation (5) in the main paper uses weight $\alpha(y, y')$ to mitigate the imbalance of class distribution. Among a mini-batch $\mathcal{B}$, the number C_S of pairs with the same label is far less than the number C_D of different labels. We therefore introduce a weight $\alpha(y, y')$ given by

$$\alpha(y, y') = \begin{cases} C_\text{D}/(C_\text{S} + C_\text{D}), & \text{for } y = y' \\ C_\text{S}/(C_\text{S} + C_\text{D}), & \text{otherwise} \end{cases}. \tag{3}$$

2.3. Implementation of k-means and PCA

We use the ResNet-18 backbone to compute feature map $F \in \mathbb{R}^{d\times h\times w}$ from all images in the training set. Let $\mathcal{F}$ denote the set of all features $f_{pq} \in \mathbb{R}^d$ in F ($p = 1, \dots, h$ and $q = 1, \dots, w$) from all images (thus, $|\mathcal{F}| = N \times h \times w$). We apply k-means or PCA to $\mathcal{F}$. The cluster centers or the principal components are deemed as concepts.

Let $f_{pq} \in \mathbb{R}^d$ be features at the spatial position (p, q) in a new image, after necessary preprocessing[1]. We can

[1]PCA's features should be centered by subtracting the mean of $\mathcal{F}$.

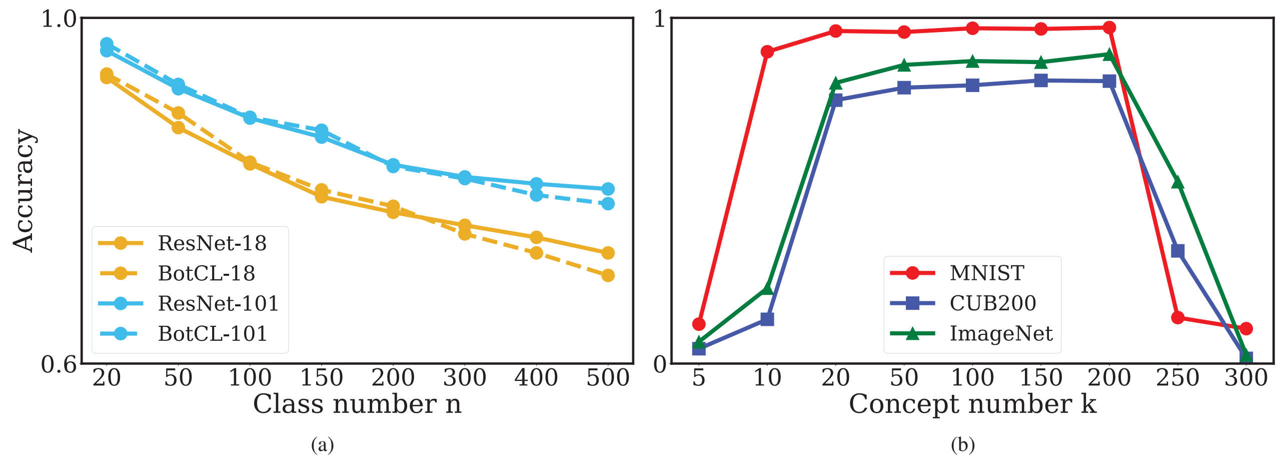


Figure 5. The relationships between the accuracy and the hyperparameter settings. (a) Number n of classes vs. accuracy. (b) Number k of concepts vs. accuracy.

calculate the soft-assignment $a_{\kappa pq}$ of f_{pq} to each concept c_κ. For k-means, we used

$$a_{\kappa pq} = e^{-\|f_{pq} - c_\kappa\|}. \tag{4}$$

For PCA, we adopt the absolute value of the cosine similarity, given by

$$a_{\kappa pq} = \text{abs}(\text{sim}(f_{pq}, c_\kappa)), \tag{5}$$

where $\text{sim}(\cdot, \cdot)$ is the cosine similarity and $\text{abs}(\cdot)$ give the absolute value.

We aggregate $a_{\kappa pq}$ for all spatial positions to form attention map $a_\kappa \in \mathbb{R}^l$. Similarly to BotCL, we summarize the presence of each concept into concept activation t_κ by reducing the spatial dimension of a_κ as

$$t_\kappa = \tanh\left(\sum_{pq} a_{\kappa pq}\right). \tag{6}$$

Also, the classifier is learned from the concept activations computed for all images in the training set.

2.4. Numbers of Classes and Concepts

In Figure 5a, we evaluate the classification performance of BotCL on subsets of ImageNet with a different number n of classes while the number k of concepts is fixed at 50). BotCL has a competitive performance when n is less than 200, compared to the ResNet baseline. However, BotCL suffers from a performance drop when n is larger than 200, which means BotCL is more suitable for small- and middle-sized tasks.

This performance drop may be relieved by increasing k, as indicated in Figure 5b, where we give the relationship between the number k of concepts and the classification accuracy (with n fixed at 10 for MNIST; 50 for CUB200 and ImageNet).

On the one hand, a large k (when $k \leq 200$) can help improve the performance. The best performance for MNIST, CUB200, and ImageNet is achieved when $k = 100$, $k = 150$, and $k = 100$, respectively. This implies that k should be tuned for each dataset to achieve the best classification accuracy. However, training fails when $k \geq 300$. This is a drawback of BotCL.

On the other hand, k is directly related to the granularity of the learned concepts. That is, a larger k tends to learn finer-grained concepts, while a smaller k leads to coarse-grained ones. Therefore, the choice of k should be decided by jointly considering the actual needs of accuracy as well as the concept granularity.

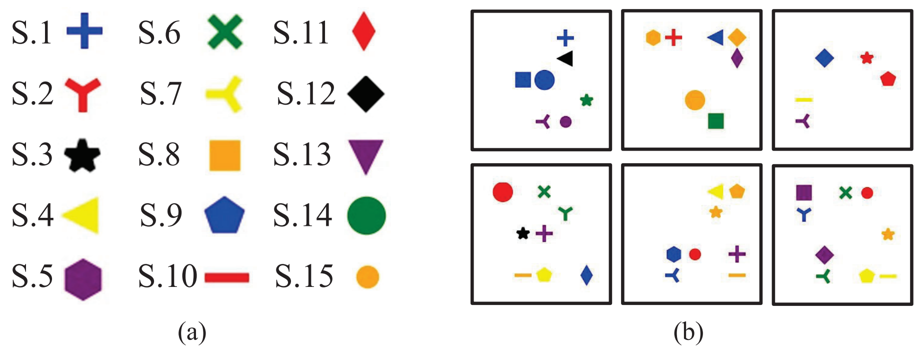


Figure 6. Generation of the Synthetic dataset. (a) Defined shapes from `S.1` to `S.15`, where (`S.1` to `S.5`) are the shapes-of-interest, while (`S.6` to `S.15`) are noises. (b) Data samples.

Table 1. Selected combinations of shapes-of-interest for the Synthetic dataset. "$\sim$" denotes NOT, "xor" denotes exclusive OR, "$+$" denotes OR, and "$\cdot$" denotes AND. For example, ω_1 presents in an image when the image does not contain both `S.1` and `S.3` or contains `S.4`.

Label	Definition
ω_1	$\sim(\texttt{S.1}\cdot\texttt{S.3})+\texttt{S.4}$
ω_2	$\texttt{S.2}+\texttt{S.3}+\texttt{S.5}$
ω_3	$\texttt{S.2}\cdot\texttt{S.3}+\texttt{S.4}\cdot\texttt{S.5}$
ω_4	`S.2` xor `S.3`
ω_5	$\texttt{S.2}+\texttt{S.5}$
ω_6	$\sim(\texttt{S.1}+\texttt{S.4})+\texttt{S.5}$
ω_7	$(\texttt{S.2}\cdot\texttt{S.3})$ xor `S.5`
ω_8	$\texttt{S.1}\cdot\texttt{S.5}+\texttt{S.2}$
ω_9	`S.3`
ω_{10}	$(\texttt{S.1}\cdot\texttt{S.2})$ xor `S.4`
ω_{11}	$\sim(\texttt{S.3}+\texttt{S.5})$
ω_{12}	$\texttt{S.1}+\texttt{S.4}+\texttt{S.5}$
ω_{13}	`S.2` xor `S.3`
ω_{14}	$\sim(\texttt{S.1}\cdot\texttt{S.5}+\texttt{S.4})$
ω_{15}	`S.4` xor `S.5`

3. Details of the Synthetic Dataset

3.1. Generation

For evaluating the performance of concept discovery, we regenerate the Synthetic dataset using the official code from ConceptSHAP [13] (as the Synthetic dataset is not directly provided). As shown in Figure 6a, there are 15 different shapes (from `S.1` to `S.15`) in this dataset. The first 5 shapes (`S.1` to `S.5`) are selected as the shapes-of-interest, and the other 10 shapes are noises. As shown Table 1, 15 different combinations of the shapes-of-interest form 15 classes. The color of the shapes is randomly picked from 'green', 'red', 'blue', 'black', 'orange', 'purple', and 'yellow'. The positions of the shapes are constrained not to overlap each other. For this, we divide an image into a 7×7 grid (which coincides ResNet's grid corresponding to F) and place a single shape in a block. We show some samples in Figure 6b.

3.2. Quantitative Metrics

We denote a set of N_{E} test images as $\mathcal{X} = \{x_i | i = 1, \dots, N_{\mathrm{E}}\}$ and a set of k learned concepts as $\mathcal{C} = \{\kappa | \kappa = 1, \dots, k\}$. For each test sample $x \in \mathcal{X}$, we denote the ground-truth position of each shapes-of-interest $\texttt{S.}j$ as s_j, which is a set of pixels inside the block (in the original image size). Meanwhile, we also define the area of each concept. For BotCL, k-means, and PCA, the spatial position of each concept is given by a_κ. We apply thresholding to a_κ to spot the concept. We denote the set of pixels whose attention value is larger than the threshold $\beta = 0.2$ by $\bar{a}_\kappa$. For ACE [5], $\bar{a}_\kappa$ includes all pixels in the super-pixels corresponding to the concept.

We first define $h_{j\kappa}$ for shape $\texttt{S.}j$ and concept κ, which represents if κ overlaps $\texttt{S.}j$, as

$$h_{j\kappa} = \begin{cases} 1, & |s_j \cap a_\kappa| / |s_j| > \gamma \\ 0, & \text{otherwise} \end{cases} . \tag{7}$$

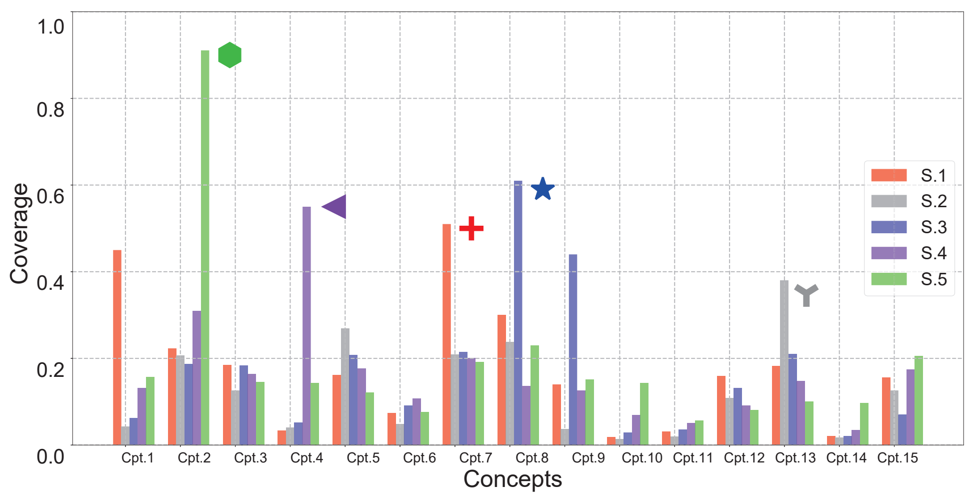
where $\cap$ is the intersection, and γ is a predefined threshold ($\gamma = 0.9$ in our setting). Note that we do not use IoU, as a single concept can cover multiple shapes. For example, one of the noise shapes (`S.6`–`S.15`) can be covered by a concept that also covers one of shapes-of-interest when the noise shape co-occurs with the shape-of-interest. In this case, the area of the concept is large, but this does not necessarily mean the discovered concept is inferior as the noise shapes are irrelevant to the target classification task. We thus design another metric named Purity to evaluate the practical purity of the concept, which is detailed later in this section.

The coverage of s by concept κ is then given by

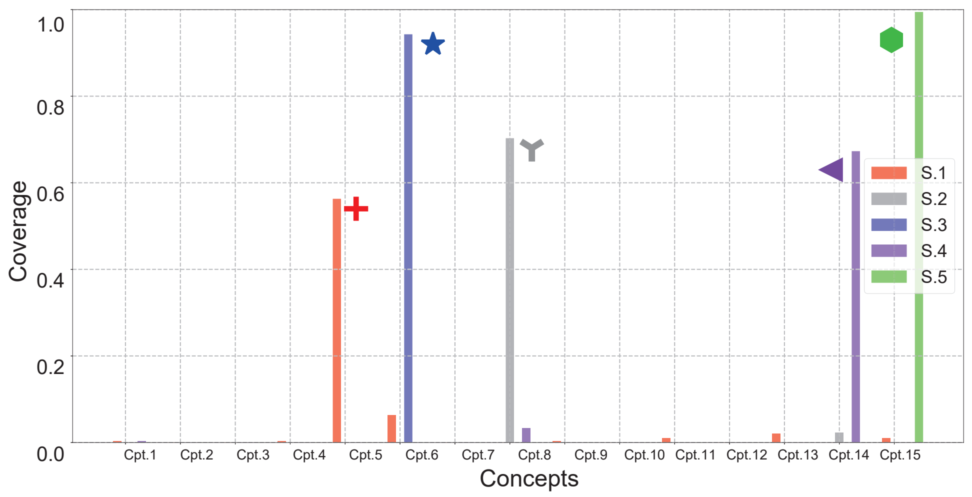
$$\text{Coverage}_{s\kappa} = \mathbb{E}[h_{s\kappa}], \tag{8}$$

which is computed over all images in $\mathcal{S}$ who contain s.

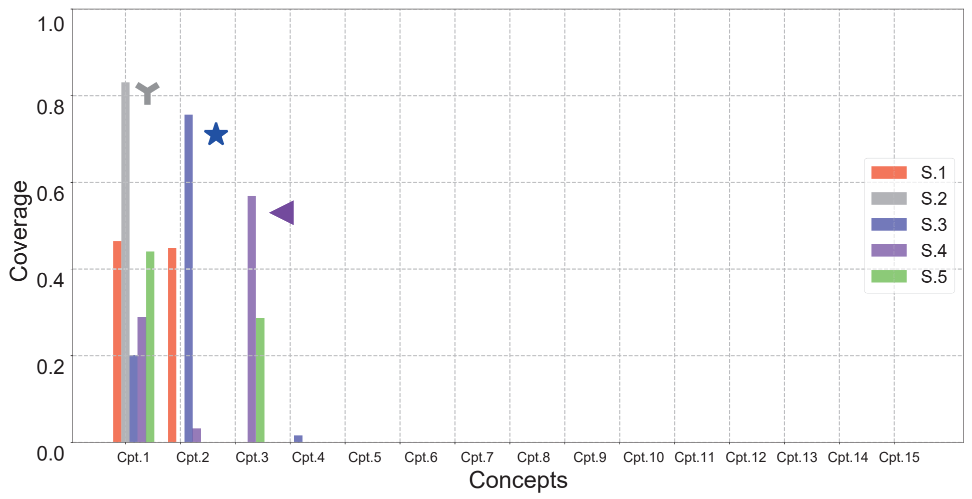
Similarly to [13], we associate each of the shapes-of-interest to one of the concepts for evaluation. Let $\mathcal{A}$ denote a set of pairs of a shape-of-interest and a concept, *i.e.*,



(a) ACE



(b) k-means



(c) PCA

Figure 7. Coverage$_{s\kappa}$ (the concept associated with each of the five shapes is marked).

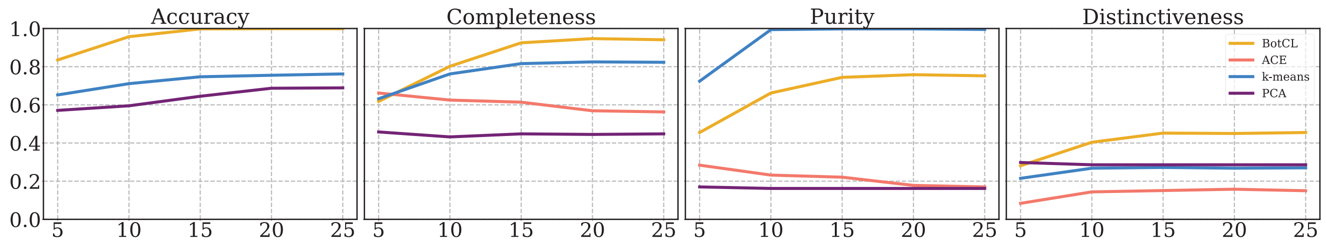


Figure 8. The impact of concept number k on BotCL, ACE, k-means, and PCA to classification accuracy, Completeness, Purity, ad Distinctiveness. Note that the classification accuracy of ACE is not shown because ACE is a post-hoc method and does not do classification by itself.

$\mathcal{A} = \{(\texttt{S}.j, \kappa_j) | j = 1, \dots, 5\}$, where $\kappa_j \in \mathcal{C}$. We can find optimal $\mathcal{A}$ by

$$\mathcal{A}^* = \text{argmax}_{\mathcal{A}} \sum_{(s,\kappa)\in\mathcal{A}} \text{Coverage}_{s\kappa}. \tag{9}$$

Note that in this maximization, only concepts in $\mathcal{A}$ are the variables and the shapes are fixed.

Based on this association, three metrics are defined to evaluate the concept discovery performance.

- **Completeness**: The most important quality of a concept is whether it has the ability to capture the associated shape completely. This can be given by

$$\text{Completeness} = \frac{1}{|\mathcal{A}^*|} \sum_{(s,\kappa)\in\mathcal{A}^*} \text{Coverage}_{s\kappa} \tag{10}$$

- **Purity**: We also expect one learned concept to be pure; that is, a concept should only cover the associated shape but not the other shapes-of-interest. Thus, we define Purity as

$$\text{Purity} = \frac{1}{|\mathcal{A}^*|} \sum_{(s,\kappa)\in\mathcal{A}^*} \frac{\text{Coverage}_{s\kappa}}{\sum_{s'} \text{Coverage}_{s'\kappa}}, \tag{11}$$

  where the summation in the denominator is computed over all shapes-in-interest.

- **Distinctiveness**: We designed BotCL so that the discovered concepts are distinctive. That is, any pair of concepts should cover different sets of shapes. We thus define distinctiveness as

$$\text{Distinctiveness} = \frac{1}{5|\mathcal{O}|} \sum_{(\kappa,\kappa')\in\mathcal{O}} \sum_{s} |\text{Coverage}_{s\kappa} - \text{Coverage}_{s'\kappa'}|, \tag{12}$$

  where $\mathcal{O}$ is the set of all possible pairs of concepts in $\mathcal{A}^*$ and the second summation is computed over all shapes-in-interest.

3.3. Coverage of ACE, k-means, and PCA

Figure 7 shows Coverage of ACE [5], k-means, and PCA (with $k = 15$). We can observe for ACE that, although some concepts tend to be dominated by one shape (*e.g.*, `Cpt.1` captures `S.5`), most of the concepts are less discriminate. For k-means, one concept captures only one shape, which leads to high Purity. However, the completeness is not as good as BotCL (refer to Figure 6b of the main paper). For example, Coverage `Cpt.5` over `S.1` is less than 0.6. In addition, PCA does not extract enough meaningful concepts.

3.4. Impact of Number k of Concepts

As shown in Figure 8, we can observe that BotCL outperforms others regardless of k in all metrics except Purity. When $0 \leq k \leq 15$, all metrics mostly improve with k. However, a larger k harms Completeness and Purity of ACE and the Distinctiveness of PCA. When $k > 15$, there are no obvious changes for all methods on all metrics. Interestingly, k-means achieves the best Purity for any k. This means that features sufficiently discriminate different shapes. However, its performance over other metrics is mostly much lower than BotCL's.

4. Details on User Study

4.1. Design of user study

Designing a user study for evaluating the interpretability of unsupervised concepts is not trivial. One straightforward way can be to ask multiple participants to write a description for each concept by reviewing *e.g.*, the top-10 activated samples, but this approach poses an extra challenge in comparing free-form descriptions. Therefore, we decided to provide a vocabulary for each dataset so that the participants could choose some terms from it.

Table 2 shows our predefined vocabularies for MNIST and CUB200. For MNIST, we set the number k of concepts to 20, but only 8 of them are activated, and the others are never activated as shown in Figure 1b. Therefore, we only show the most activated images of these 8 concepts to the participants. The vocabulary consists of two groups, *position* and *shape*. These groups are combinatorial; the participants choose one from each to describe the concept. We found that some concepts cover two different elements of the digits, so we allow the participants to specify two pairs of position and shape. For example, a participant may choose *upper* and *a horizontal line* as well as *lower* and *a (part of) curve* for `Cpt.11` (refer to Figure 1b, as it involves two highlighted regions. When no consistent concept can be found in the provided samples, participants can choose *None of them*. For CUB200, all 20 concepts learned from a subset with $n = 50$ classes are presented. The vocabulary is defined based on the terms related to birds, falling into five groups (i) *Body Part*, (ii) *Color*, (iii) *Texture*, (iv) *Action*, and (v) *Background*. Each group requires to choose one term. Otherwise, a participant can choose *None of them* when no consistent concept can be found. We provide the screenshot of our user interface in Figure 9 and 10.

4.2. Metrics

We designed four metrics to evaluate learned concepts based on the user study:

- **Concept discovery rate** (CDR): This metric is the ratio of participants who can successfully find a meaningful concept in given samples, *i.e.*, the ratio of participants who selected terms other than *None of them*. This metric directly indicates how human-understandable the learned concepts at a conscious level.

- **Concept consistency** (CC): This metric involves the consistency of responses of a pair of participants for one concept, measuring inter-participant differences in the perception of a concept. Let R_{gi} denote participant i's response on group g, which is one of the terms in the group g. CC is formulated as :

$$\text{CC} = \sum_{g \in \mathcal{G}} w_g r_g \frac{1}{|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} \mathbb{I}(R_{gi}, R_{gj}), \tag{13}$$

  where $\mathbb{I}$ is the indicator function that gives 1 when $R_{gi} = R_{gj}$, and 0 otherwise. $\mathcal{G}$ is the set of all groups, and $\mathcal{P}$ is the set of all possible pairs of participants. For MNIST, we expand the groups by making all possible combinations of positions and shapes because it is more natural to see the *position* group as modifiers. Therefore, for MNIST, $|\mathcal{G}| = 1$ and this single group contains $3 \times 8 = 24$ terms. We introduce w_g to compensate for the imbalance among the number of times, one term of each group is selected so that a group that is used many times can contribute more to the final score. w_g is the ratio of times in which one term in group g is selected overall responses. r_g is a discount factor for *None of Them*. Let η be the number of all pairs of participants and η'_g the number of pairs whose responses for group g are both non-*None of Them*. We define the discount factor as $r_g = 1 - \eta'_g/\eta$.

- **Mutual information between concepts** (MIC): This metric measures the similarity of the response distribution over all possible pairs of concepts. Letting H denote the concatenation of histograms H_g for all group g and H' is the same concatenated histogram, but for a different concept, it can be formulated as follows:

$$\text{MIC} = \text{MI}(H, H'), \tag{14}$$

  where MI gives the mutual information between H and H'. Note that for MIC, the statistics (the mean and standard deviation) are computed over all possible pairs of concepts, whereas for the other three metrics, they are computed over all concepts.

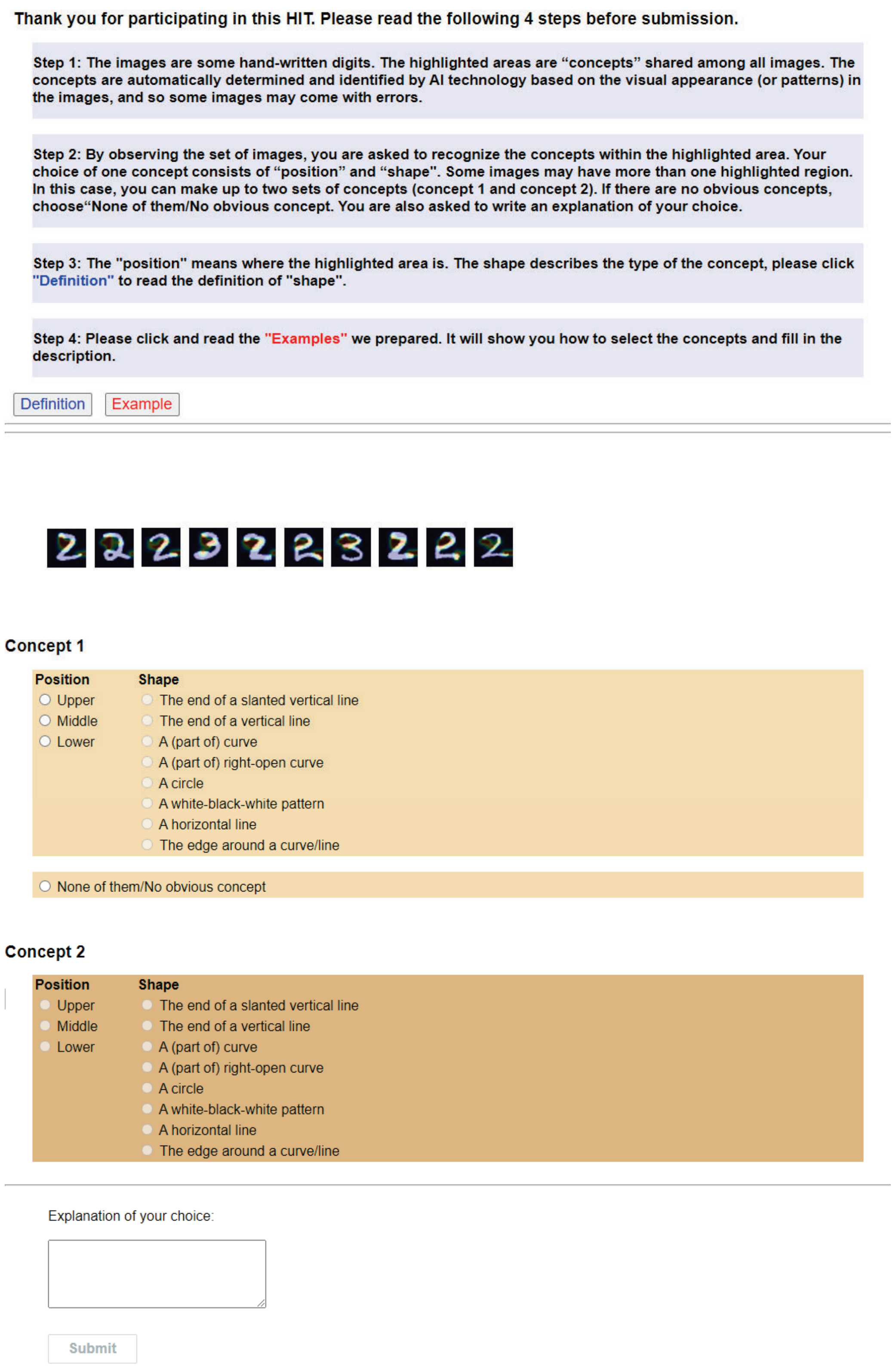
For comparison, we conducted an extra round of user study with manually labeled concepts and random concepts. To be consistent with BotCL's setting, we used the same number of concepts (8 for MNIST and 20 for CUB200) as well as the number of participants (20 for MNIST and 30 for CUB200). For manually labeled concepts, we picked out a (combination of) terms from our vocabulary to make a concept and selected 10 images that contained the concept. We then manually annotated the region corresponding to the concept in each image. This renders a certain cap for each metric. For random concepts, we randomly selected 10 samples for each concept and randomly generated highlights for each sample; therefore, there barely be a consistent concept within the samples. Figures 13–16 show some examples of manually labeled and random concepts for both MNIST and CUB200.

Table 2. Vocabulary used in the user study.

Dataset	Group	Vocabulary
MNIST	Position (3)	upper, middle, lower
	Shape (8)	the end of a slanted vertical line, the end of a vertical line, a (part of) curve, a (part of) right-open curve, a circle, a white-black-white pattern, a horizontal line, the edge around a curve/line
CUB200	Body Part (9)	head, wing, leg, beak, crawl, breast, tail, neck, back
	Color (10)	red, grey, beige, black, yellow, brown, white, blue, green, colorful
	Texture (2)	striped, spotted
	Action (4)	flying, swimming, climbing, perching
	Background (5)	sea, tree, sky, grass, land

We show the distributions of participants' answers in Figure 11 and 12. For MNIST, we can find that most concepts are recognized to be meaningful. The participants tend to choose the same term for one concept, *e.g.*, the option for `Cpt.8` mostly described by *lower* and *an end of a slanted vertical line*. However, we also observe that `Cpt.5` cannot be identified by most of the participants, as its highlighted regions are too weak and hardly noticeable as shown in Figure 1b. For CUB200, we show the distribution of each group in the first five columns, and the last column shows the number of participants who selected *None of them*. We find that most of the learned concepts in CUB200 are meaningful, as the number of *None of them* is small for most concepts. Participants' responses are mostly distributed in Body Part (especially *Wing* and *Leg*), *Color* (*Black*), and *Action* (*Perching*).

From this user study, we would conclude that the concepts learned by BotCL are recognizable, individually consistent, and mutually distinct for humans, comparable with manually labeled concepts, which means that BotCL can potentially apply to a wide range of applications that require interpretability.



Thank you for participating in this HIT. Please read the following 4 steps before submission.

Step 1: The images are some hand-written digits. The highlighted areas are “concepts” shared among all images. The concepts are automatically determined and identified by AI technology based on the visual appearance (or patterns) in the images, and so some images may come with errors.

Step 2: By observing the set of images, you are asked to recognize the concepts within the highlighted area. Your choice of one concept consists of “position” and “shape". Some images may have more than one highlighted region. In this case, you can make up to two sets of concepts (concept 1 and concept 2). If there are no obvious concepts, choose“None of them/No obvious concept. You are also asked to write an explanation of your choice.

Step 3: The "position" means where the highlighted area is. The shape describes the type of the concept, please click "Definition" to read the definition of "shape".

Step 4: Please click and read the "Examples" we prepared. It will show you how to select the concepts and fill in the description.

Concept 1

Position	Shape
○ Upper	○ The end of a slanted vertical line
○ Middle	○ The end of a vertical line
○ Lower	○ A (part of) curve
	○ A (part of) right-open curve
	○ A circle
	○ A white-black-white pattern
	○ A horizontal line
	○ The edge around a curve/line

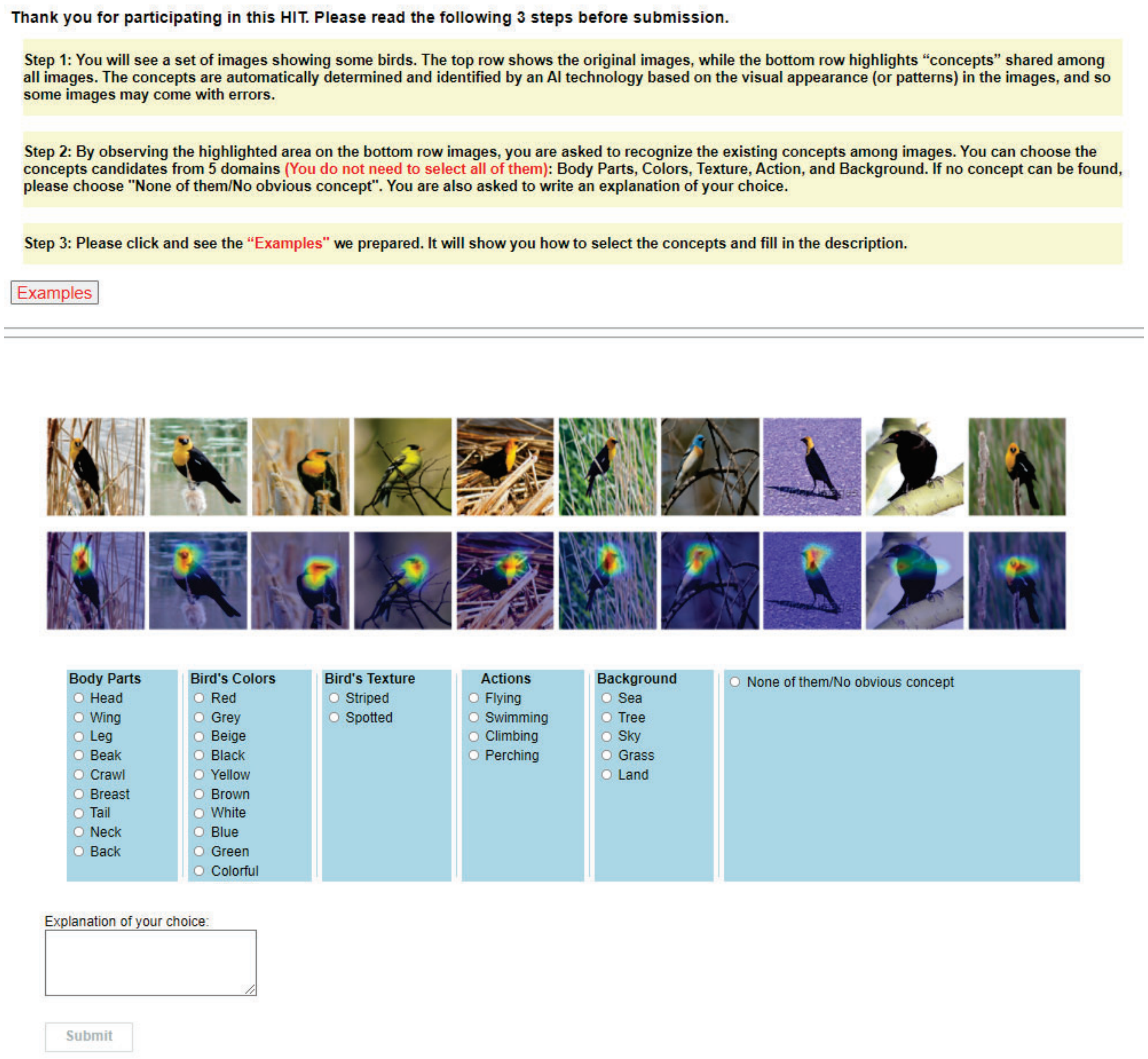
○ None of them/No obvious concept

Concept 2

Position	Shape
○ Upper	○ The end of a slanted vertical line
○ Middle	○ The end of a vertical line
○ Lower	○ A (part of) curve
	○ A (part of) right-open curve
	○ A circle
	○ A white-black-white pattern
	○ A horizontal line
	○ The edge around a curve/line

Explanation of your choice:

Figure 9. User interface of the user study for MNIST.



Thank you for participating in this HIT. Please read the following 3 steps before submission.

Step 1: You will see a set of images showing some birds. The top row shows the original images, while the bottom row highlights “concepts” shared among all images. The concepts are automatically determined and identified by an AI technology based on the visual appearance (or patterns) in the images, and so some images may come with errors.

Step 2: By observing the highlighted area on the bottom row images, you are asked to recognize the existing concepts among images. You can choose the concepts candidates from 5 domains (You do not need to select all of them): Body Parts, Colors, Texture, Action, and Background. If no concept can be found, please choose "None of them/No obvious concept". You are also asked to write an explanation of your choice.

Step 3: Please click and see the “Examples” we prepared. It will show you how to select the concepts and fill in the description.

Body Parts	Bird's Colors	Bird's Texture	Actions	Background	None of them/No obvious concept
Head	Red	Striped	Flying	Sea	
Wing	Grey	Spotted	Swimming	Tree	
Leg	Beige		Climbing	Sky	
Beak	Black		Perching	Grass	
Crawl	Yellow			Land	
Breast	Brown				
Tail	White				
Neck	Blue				
Back	Green				
	Colorful				

Explanation of your choice:

Figure 10. User interface of the user study for CUB200.

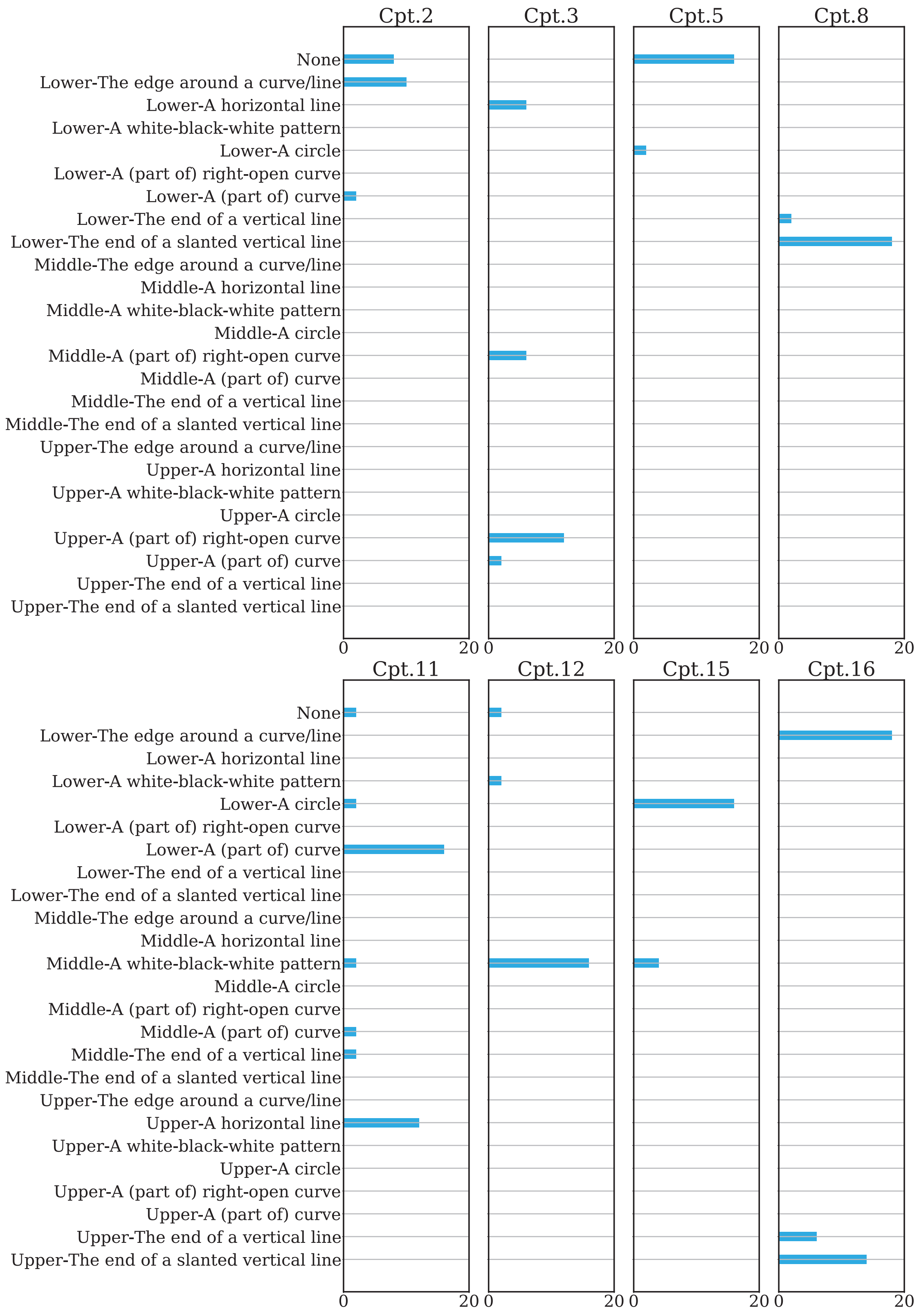


Figure 11. BotCL's distribution of responses for MNIST.

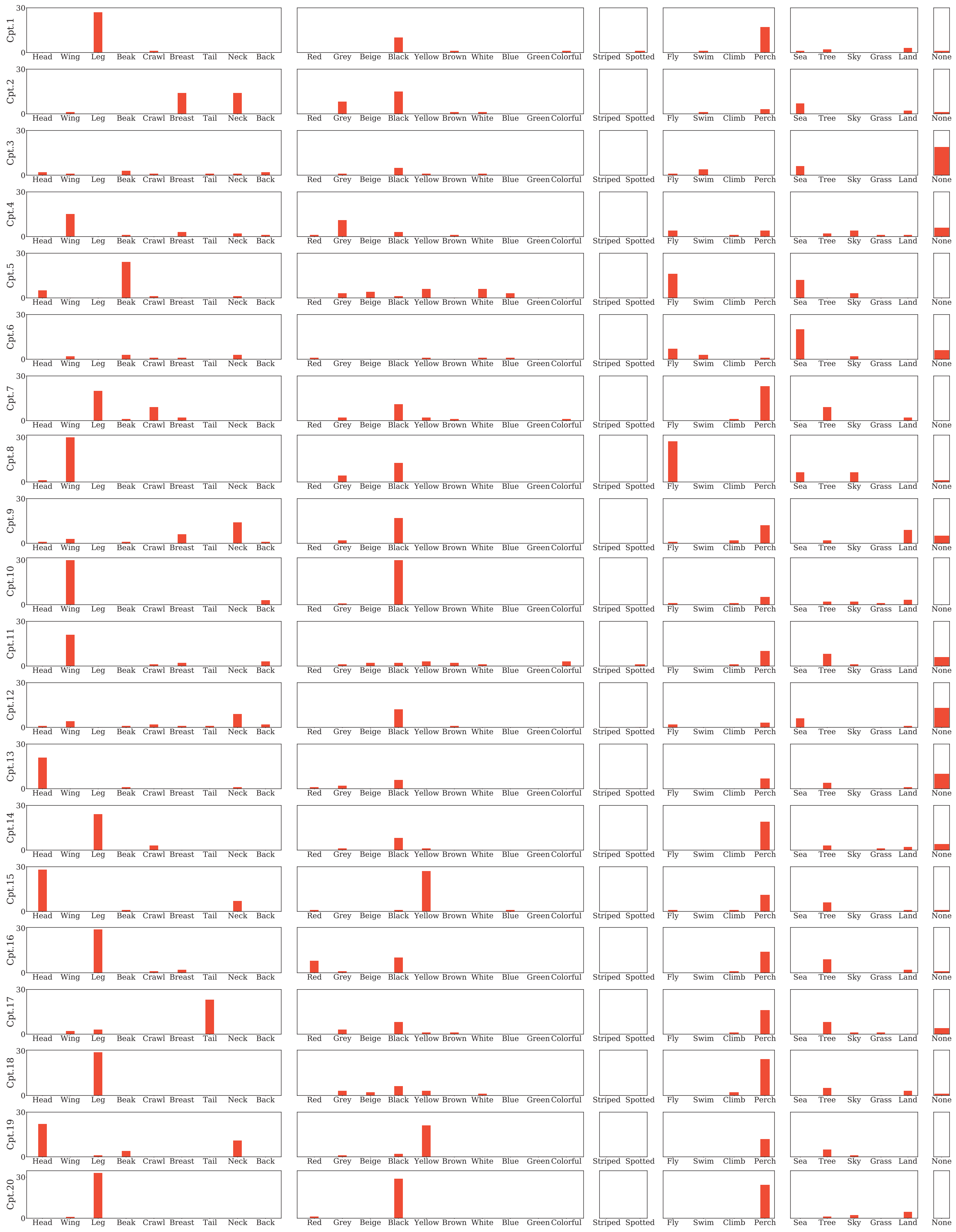


Figure 12. BotCL's distribution of responses for CUB200.

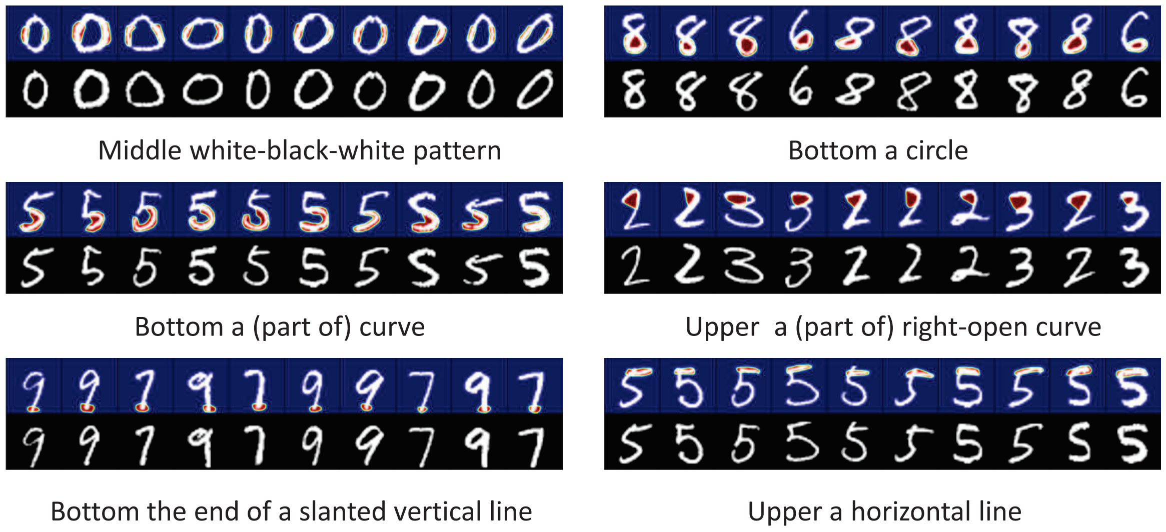

Middle white-black-white pattern
Bottom a circle
Bottom a (part of) curve
Upper a (part of) right-open curve
Bottom the end of a slanted vertical line
Upper a horizontal line

Figure 13. Examples of manually labeled concepts for MNIST.

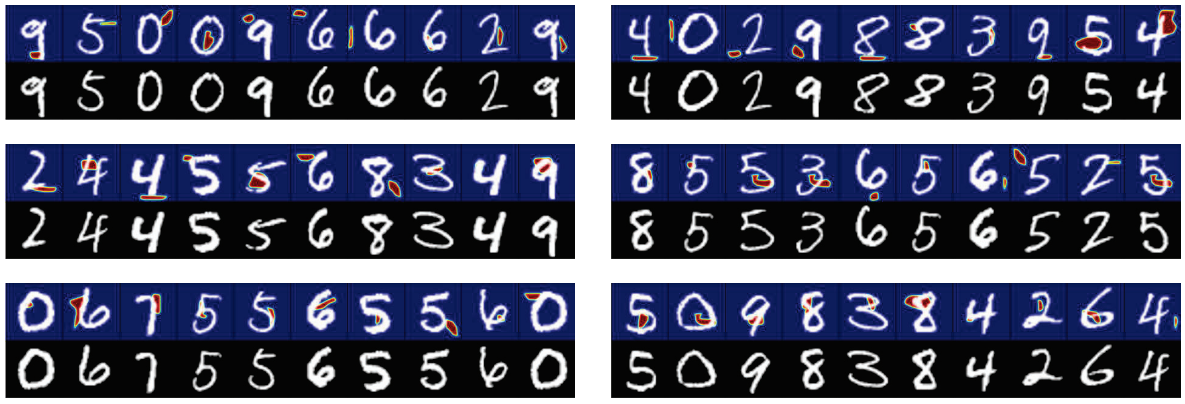


Figure 14. Examples of random concepts for MNIST.

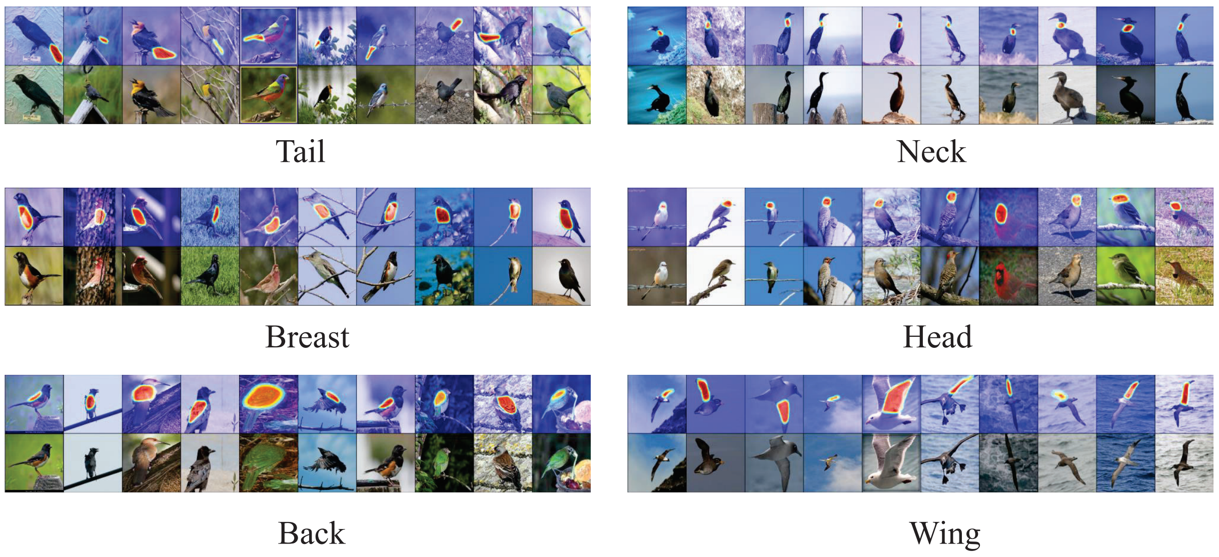


Figure 15. Examples of manually labeled concepts for CUB200.

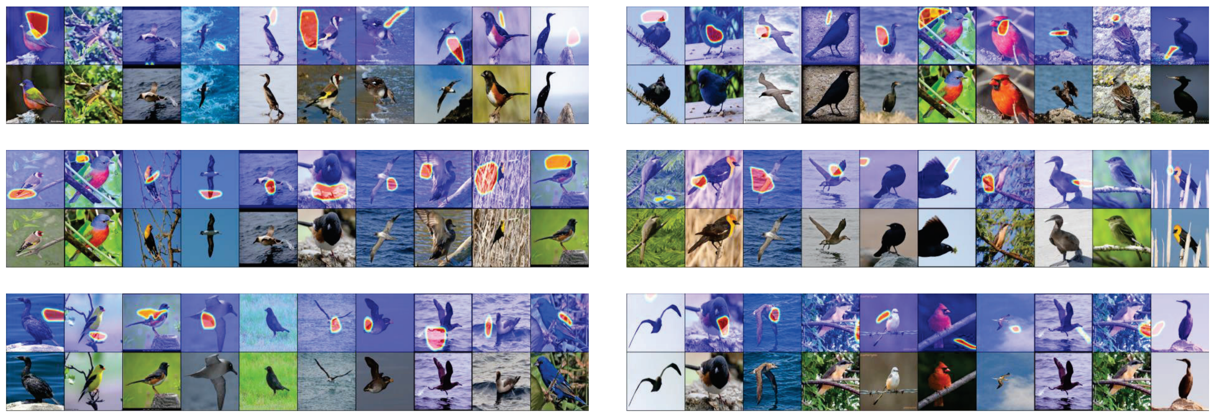


Figure 16. Examples of random concepts for CUB200.

5. Comparison to existing XAI methods

BotCL aims at learning concepts, which is completely different from per-pixel importance-based XAI methods. Therefore, the explainability scores tailored for these XAI methods are not the main concern of this paper. Yet, comparing BotCL with major XAI methods gives strong evidence of its explainability. For this comparison, the attention of each concept can be merged into an overall explanation $\bar{a}$ by the weighted sum as

$$\bar{a} = \frac{1}{k} \sum_{\kappa} a_{\kappa} z_{\omega\kappa}, \tag{15}$$

where ω is the ground-truth class. We adopt four evaluation metrics, including Insertion area under curve (IAUC) and deletion area under curve (DAUC) are the metrics designed in [7], Stability [1], and Infidelity [13].

IAUC is calculated by gradually adding pixels (in the order of importance) to a blank image and seeing how the prediction confidence evolves. The prediction confidence should rise quickly if the pixel-adding process is guided by an explanation that well understands the model and thus can point out the most important pixels. In contrast, **DAUC** is calculated by gradually removing pixels (in the order of importance) from the original image. Similarly, the prediction confidence should drop quickly if the pixel-removing process is guided by an explanation.

Stability is quantified by Lipschitz estimation to measure how stable the explanation method performs when the input is perturbed with minor noises. It can be formulated as follows:

$$\text{Stability} = \frac{\|\mathcal{E}_{\gamma}(x) - \mathcal{E}_{\gamma}(x')\|_2}{\|x - x'\|_2}, \tag{16}$$

where x is the original input and x' is the perturbed input. γ is a model (*i.e.*, a composition of feature extractor Ψ and classifier f, and $\mathcal{E}_{\gamma}$ is the function to generate an explanation of model γ. Adding minor white noise to the input image should not have a significant impact on the prediction result. However, the explanation may change a lot if the method is instability.

Infidelity measures the consistency between input perturbations and consequent significant explanation changes. It is formulated as follows:

$$\text{Infidelity} = \mathbb{E}_{I \sim \mu_I}[(I^{\top}\mathcal{E}_{\gamma}(x) - (\gamma(x) - \gamma(x - I)))^2], \tag{17}$$

where I is a significant perturbation to the input with one probability measure μ_I, and the variables (*i.e.*, I and $\mathcal{E}_{\gamma}$) are vectorized if necessary. The paper [13] provides multiple options for μ_I, and we chose $\mu_I = \mathcal{N}(0, \sigma^2)$.

In addition, the explainability of the existing XAI methods is evaluated with the baseline ResNet [6] model, which uses a single FC as the classifier, while our results are obtained on BotCL, which uses the same ResNet model (without the FC classifier) as the backbone. In Table 3, we can see that BotCL achieves the best scores in stability and infidelity, and is among the best for IAUC/DAUC. BotCL is slightly worse than the best results in IAUC is that BotCL requires the activations of enough number concepts to lead to the correct classification, for which more pixels are necessary. Similarly, BotCL works well even if some concepts are not activated. More pixels need to be masked to change the output, which implies BotCL's robustness.

Table 3. Evaluation of BotCL and existing XAI methods using explainability metrics.

	CUB200				ImageNet			
Methods	Stability ↓	Infidelity ↓	IAUC ↑	DAUC ↓	Stability ↓	Infidelity ↓	IAUC ↑	DAUC ↓
LIME [8]	0.175	0.150	0.664	0.133	0.211	0.398	0.624	0.154
CAM [14]	0.170	0.138	0.695	0.114	0.208	0.372	0.678	0.135
GradCAM [9]	0.155	0.142	0.712	0.110	0.180	0.358	0.682	0.130
GradCAM++ [2]	0.168	0.135	**0.731**	**0.099**	0.188	0.360	0.687	0.121
Score-CAM [11]	0.160	0.122	0.725	0.102	0.176	0.355	**0.697**	**0.118**
SS-CAM [10]	0.166	0.130	0.698	0.109	0.191	0.377	0.675	0.133
BotCL	**0.102**	**0.051**	0.718	0.105	**0.125**	**0.341**	0.680	0.131

References

[1] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *NeurIPS*, 2018.

[2] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *WACV*, pages 839–847, 2018.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[4] Li Deng. The mnist database of handwritten digit images for machine learning research. *Signal Processing Magazine*, 29(6):141–142, 2012.

[5] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. Towards automatic concept-based explanations. *NeurIPS*, 2019.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[7] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. *BMVC*, 2018.

[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ” Why should i trust you?” Explaining the predictions of any classifier. In *ACM SIGKDD*, pages 1135–1144, 2016.

[9] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *CVPR*, pages 618–626, 2017.

[10] Haofan Wang, Rakshit Naidu, Joy Michael, and Soumya Snigdha Kundu. SS-CAM: Smoothed Score-CAM for sharper visual feature localization. *arXiv preprint arXiv:2006.14255*, 2020.

[11] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-CAM: Score-weighted visual explanations for convolutional neural networks. In *CVPR workshops*, pages 24–25, 2020.

[12] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD birds 200. 2010.

[13] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. In *NeurIPS*, volume 33, pages 20554–20565, 2020.

[14] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016.