Learning Bottleneck Concepts in Image Classification

Vishwanath Hurakadli AI20BTECH11023
Sujal AI20BTECH11020

Abstract

Explainable AI is crucial for understanding deep neural networks. While current methods provide per-pixel relevance, they may require expert knowledge. This paper (Wang et al., 2023) introduces Bottleneck Concept Learner (BotCL), which represents images based on concepts learned during training without explicit supervision. BotCL uses self-supervision and tailored regularizers to make learned concepts human-understandable. Tests on imaThis document provides a basic paper template and submission guidelines. Abstracts must be a single paragraph, ideally between 4–6 sentences long. Gross violations will trigger classification tasks show BotCL's potential for improving neural network interpretability corrections at the camera-ready phase.

1. Introduction

Explaining deep neural network (DNN) behavior in medical applications and addressing biases is challenging. Current post-hoc explanations use relevance maps but lack semantic depth, requiring expert interpretation. A concept-based framework offers higher-level relationships by associating decisions with a set of concepts in the image. The concept bottleneck structure, using concepts' presence/absence, is popular, but designing task-specific concepts is difficult. The proposed Bottleneck Concept Learner (BotCL) discovers concepts and learns the classifier simultaneously, representing images by concept existence. BotCL employs self-supervision and constraints for concept quality, utilizing a slot attention-based mechanism. The approach enhances interpretability and achieves concept discovery without explicit supervision.

Contribution. We tried to address a problem mentioned by main authors of the paper to identify the number of concepts in any dataset using conceptnet api and further processing of obtained concepts We test this architecture on datasets not tested in main paper including FashionMNIST

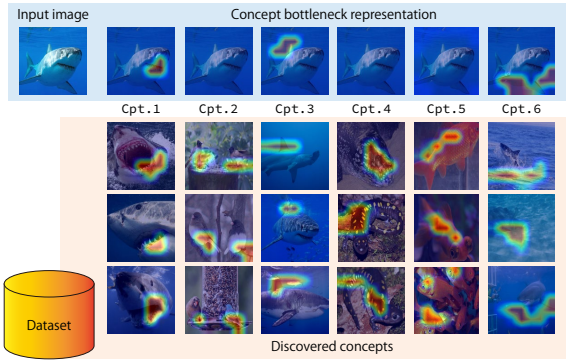

Figure 1. Examples of concepts discovered by BotCL in ImageNet (Deng et al., 2009) and concepts in the input image. BotCL automatically discovers a set of concepts optimized for the target task and represents an image solely with the presence/absence of concepts.

2. Model

Given a dataset $\mathcal{D} = \{(x_i, y_i)|i = 1, 2, \ldots, N\}$, 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 2 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.

2.1. Concept labels

To identify the number of concepts for each datasets using its label, we are using ConceptNet a semantic network to

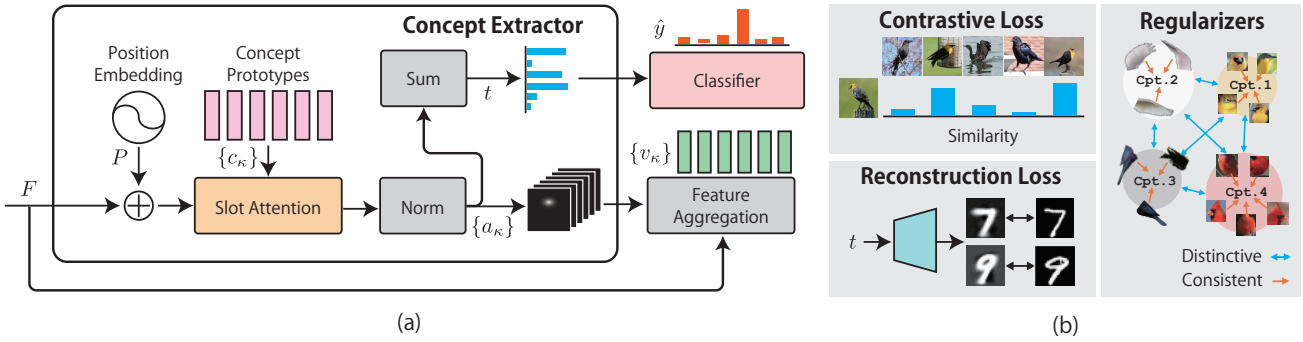

Figure 2. (a) The model pipeline. (b) Self-supervision and regularizers.

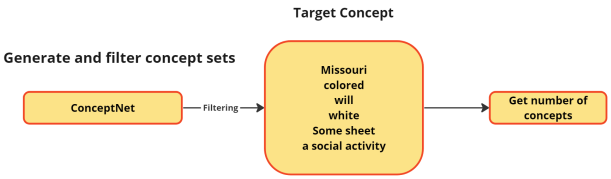

Figure 3. Enter Caption

create word embeddings – representations of word meanings as vectors, similar to word2vec, GloVe, or fastText, but better. Inspired from Label-Free CBM (Tuomas Oikarinen, 2023) to use set of concepts as text embeddings, instead to considering embeddings, we can get information about concepts present in each class in datasets. Using class labels to get list of concepts present in images which can also help in labelling the trained concepts. So K can be number of identified and filtered concepts.

2.2. Concept Extractor

Concept extractor uses slot attention (Locatello et al., 2020; Li et al., 2021)-based mechanism to discover visual concepts in $\mathcal{D}$. We first add position embedding P to feature map F to retain the spatial information, that is, $F' = F + P$. 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 ϕ 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.

2.3. 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.

2.4. Classifier

We use a single FC layer without a bias term as the classifier, and concept activation $t = (t_1, \ldots, t_k)^\top$ is the only input, serving as the concept bottleneck (Koh et al., 2020). 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.

3. Training

3.1. 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 (Alvarez-Melis & Jaakkola, 2018) uses an autoencoder-like structure for learning better

representation. We assume this structure works well when visual elements are strongly tied with the position[1] 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 **??**, 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 (Chen et al., 2020; He et al., 2020).

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 (, $y = y'$ 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

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

3.2. 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.

3.3. 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.

3.4. 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.

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
LightGrey Baseline	0.731	0.786	0.988	0.999
k-means* (Yeh et al., 2020)	0.063	0.427	0.781	0.747
PCA* (Yeh et al., 2020)	0.044	0.139	0.653	0.645
SENN (Alvarez-Melis & Jaakkola, 2018)	0.642	0.673	**0.985**	0.984
ProtoPNet (Chen et al., 2019)	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**

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

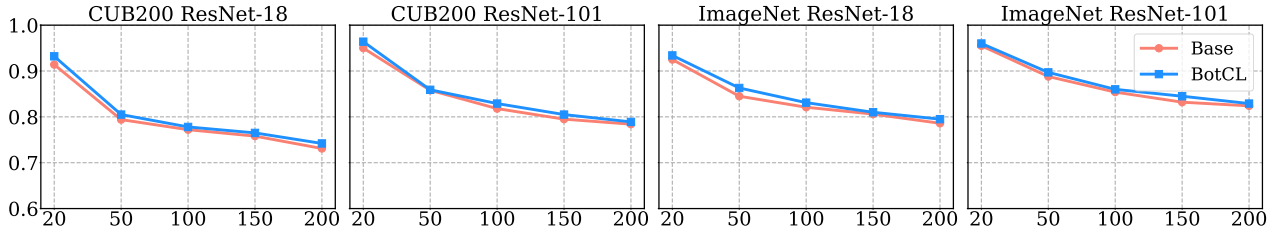

Figure 4. Classification accuracy vs. the number of classes. We used subsets of CUB200 and ImageNet with $k = 50$ and ResNet-18 and ResNet-101 backbones.

4. Results

4.1. Experimental Settings

We evaluate BotCL on MNIST (Deng, 2012), CUB200 (Welinder et al., 2010), and ImageNet (Deng et al., 2009). For evaluating discovered concepts, we regenerated a synthetic shape dataset (Synthetic) (Yeh et al., 2020).

For MNIST, we applied the same networks as (Alvarez-Melis & Jaakkola, 2018) for the backbone and the concept decoder. For CUB200 (same data split as (Koh et al., 2020)) and ImageNet, we used pre-trained ResNet (He et al., 2016) 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 (Yeh et al., 2020), 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 $\lambda_{\text{R}} = 0.1$.

4.2. 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 (Yeh et al., 2020) and state-of-the-art concept-based models. Table 1 summarized the results. BotCL with contrastive loss ($\text{BotCL}_{\text{Cont}}$) 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 ($\text{BotCL}_{\text{Rec}}$) shows a performance drop over CUB200, ImageNet, and Synthetic, while it outperforms $\text{BotCL}_{\text{Cont}}$ 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 4 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 $n > 200$ for larger n and different k's).

4.3. Experiments

We are using ConceptNet to generate concept labels and filtering generated labels to get most appropriate labels. Using the number of generated concept labels using class labels as approximation for number of concepts to be trained, we can get better approximation in this case. So for CUB200 dataset we get aproximately 32 concepts after filtering which includes Missouri, colored, will, white, Some sheet,a social activity, capped. Similarly for CIFAR10 dataset we got approx. 121 as number of concepts. This indicates that in CIFAR10 each class is having unique and non-overlapping concepts but in CUB200 each classes are of birds and most of them have overlapping concepts thus reducing number of concepts. We also tested this architecture on Fashion-MNIST dataset and with number of concepts 20 and with reconcstrution loss we got best training accuracy of 0.8137.

Dataset	Number of Concepts
CUB200	32
CIFAR10	121
ImageNet	230

Table 2. Number of Concepts for dataset

References

Alvarez-Melis, D. and Jaakkola, T. S. Towards robust interpretability with self-explaining neural networks. *NeurIPS*, 2018.

Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., and Su, J. K. This looks like that: Deep learning for interpretable image recognition. *NeurIPS*, 32, 2019.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *ICML*, pp. 1597–1607, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9729–9738, 2020.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. Concept bottleneck models. In *ICML*, pp. 5338–5348, 2020.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, L., Wang, B., Verma, M., Nakashima, Y., Kawasaki, R., and Nagahara, H. Scouter: Slot attention-based classifier for explainable image recognition. In *ICCV*, pp. 1046–1055, 2021.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *NeurIPS*, 2020.

Tuomas Oikarinen, Subhro Das, L. M. N. T.-W. W. Label-free concept bottleneck models. 2023.

Wang, B., Li, L., Nakashima, Y., and Nagahara, H. Learning bottleneck concepts in image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-UCSD birds 200. 2010.

Yeh, C.-K., Kim, B., Arik, S., Li, C.-L., Pfister, T., and Ravikumar, P. On completeness-aware concept-based explanations in deep neural networks. In *NeurIPS*, volume 33, pp. 20554–20565, 2020.