A. Implementation Details

All experiments are conducted on a high-performance computing cluster equipped with four NVIDIA A100 80GB GPUs. The system runs on CUDA 12.4 with NVIDIA driver version 550.54.14. The deep learning framework used for implementation is PyTorch, and model training is optimized using mixed precision and multi-GPU distributed training to maximize efficiency.

A.1. Dimension Unification

The encoder and decoder are designed symmetrically, each consisting of two fully connected layers. For GoogleNet and MobileNetV2, the hidden layer has 4096 units, and the final embedding layer also has 4096 units. For ResNet50V2, both the hidden and embedding layers are reduced to 2048 units to optimize GPU memory usage and prevent out-of-memory (OOM) errors during training. Each layer is followed by a ReLU activation function to introduce non-linearity and improve representation learning.

A.2. Contrastive Learning for CAV Alignment

A.2.1. MLP Projection Head design

Here we introduce the details of implementation of the MLP Projection head in Figeure **??**. To ensure that embeddings obtained from different layers are projected into a shared semantic space, we employ a residual MLP architecture for $f(\cdot)$. This design follows two key principles:

- **Alignment Across Layers:** Since embeddings come from different autoencoders trained on separate layers, their distributions may vary. The MLP projection helps unify these representations.
- **Preserving Original Semantics:** A residual connection ensures that the transformed embeddings retain the essential concept-specific information from the original embeddings.

The function is defined as follows:

```
def projection_function(z):
    # Non-linear transformation through MLP
    projected = MLP(z)
    # Residual connection with a linear transformation
    residual = Linear(z)
    # Normalization to maintain consistent scale
    return normalize(projected + residual)
```

Here, “MLP(z)" denotes a multi-layer perceptron with LayerNorm and GELU activations, ensuring smooth and stable training. “Linear(z)" represents a simple linear layer that directly maps the input to the output space, enabling residual learning. The final normalization step ensures that the projected embeddings remain in a comparable range.

This projection function plays a crucial role in learning a unified representation for concept activation vectors across different layers.

A.3. Cross-Layer CAV Fusion

In this section, we detail the design of our fusion module and the loss function used to train the model.

A.3.1. Transformer-Based Cross-Layer CAV Fusion

To fuse CAV embeddings from different layers, we employ an attention-based fusion module. Given a set of aligned CAV embeddings

$$\{\mathbf{z}_l\}_{l=1}^{L}, \quad \mathbf{z}_l \in \mathbb{R}^d,$$

the fusion module computes a global CAV representation. In particular, positional encoding is added to the input embeddings, followed by a multi-head self-attention layer and a feed-forward network (FFN). A residual connection and layer normalization are applied at each stage. Finally, an average pooling across the layer dimension yields the global CAV:

$$\mathbf{z}_{\text{global}} = f_{\text{fuse}}(\{\mathbf{z}_l\}) = \text{OutputLayer}\left(\frac{1}{L}\sum_{l=1}^{L}\tilde{\mathbf{z}}_l\right). \tag{1}$$

The pseudo-code for the fusion module is as follows:

```
Input: CAV embeddings [batch_size, num_layers, embedding_dim]
1. Add positional encoding to each layer's embedding.
2. Compute self-attention via a multi-head attention layer.
3. Apply residual connection and layer normalization.
4. Process the result with a feed-forward network and dropout.
5. Apply a second residual connection and normalization.
6. Aggregate across layers (e.g., average pooling).
7. Transform the pooled result via a linear output layer.
Output: Global CAV [batch_size, embedding_dim]
```

A.3.2. Loss Design

The overall loss function consists of two components:

- $\mathcal{L}_{\text{var}}$: The **layer variance loss**, which enforces layer-wise consistency by minimizing the variance of TCAV scores across layers.
- $\mathcal{L}_{\text{cons}}$: The **concept center separation loss**, which ensures that the reconstructed CAVs remain semantically aligned with the original ones by enforcing a cosine-similarity constraint.

The final loss function is formulated as:

$$\mathcal{L} = \lambda_{\text{var}}\,\mathcal{L}_{\text{var}} + \lambda_{\text{cons}}\,\mathcal{L}_{\text{cons}}, \tag{2}$$

where λ_{var} and λ_{cons} are balancing weights that regulate the trade-off between cross-layer consistency and semantic preservation.

The loss computation follows these steps:

```
Input:
    - GCAV [batch_size, cav_dim]
    - Decoders for each layer
    - Class activation values per layer
    - Class examples and target labels
    - Additional model parameters (e.g., bottleneck, concept labels)

1. Update the dynamic temperature parameter.
2. For each layer:
    a. Reconstruct the CAV using the corresponding decoder.
    b. Compute cosine similarity between the reconstructed and original CAV.
    c. Compute the TCAV score using a gradient-based method.
3. Compute consistency loss as the average cosine loss over layers.
4. Compute variance loss as the variance of TCAV scores across layers.
5. Compute total loss:
       total_loss = var_weight * variance_loss + consistency_weight * consistency_loss.
6. Return total_loss.
```

These components ensure that the fusion module aggregates multi-layer information into a unified GCAV while preserving consistency and maintaining clear semantic separation between concepts.

During training, the computation of TCAV scores s_l at layer l follows the standard TCAV formulation. As described in Section **??**, we obtain the reconstructed concept vector $\tilde{\mathbf{v}}_l^c$ by decoding the globally integrated representation $\mathbf{z}_{GCAV}^c$ using the layer-specific decoder f_{decoder}^l. We then replace the original CAV with $\tilde{\mathbf{v}}_l^c$ and compute the TCAV score as:

$$TCAV_{c,k,l} = \frac{\left|\{x \in X_k : \nabla h_{l,k}\big(f_l(x)\big) \cdot \tilde{\mathbf{v}}_l^c > 0\}\right|}{|X_k|}, \tag{3}$$

where X_k denotes the set of inputs belonging to class k, $f_l(x)$ represents the activation at layer l for input x, and $h_{l,k} : \mathbb{R}^m \to \mathbb{R}$ maps these activations to the logit for class k. This formulation measures the proportion of class examples for which the directional derivative along $\tilde{\mathbf{v}}_l^c$ aligns with the classification objective.

To enable gradient-based optimization, we approximate the non-differentiable ">" operator using a sigmoid relaxation:

$$s_l^{(i)} = \mathrm{STE}\Big(\sigma\Big(-\tau\,\nabla h_{l,k}(f_l(x)) \cdot \tilde{\mathbf{v}}_l^c\Big)\Big), \tag{4}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, STE is the straight-through estimator [**?**], and τ is a temperature parameter that gradually increases during training. This progressively sharpens the approximation of the hard threshold, allowing the loss to remain differentiable while converging toward the standard TCAV formulation.

By integrating these loss components, the model learns a globally aligned concept representation that remains stable across layers while preserving the distinctiveness of individual concepts.

B. More Results

This section provides ablation studies, hyperparameter investigations, and extensive tests on high-level concepts using more images from additional datasets. We also present results from additional visualizations of TCAV and TGCAV scores across layers in different models, including GoogleNet, MobileNetV2, and ResNet50V2.

B.1. Ablation Study

Table 1 shows ablations for the "Zebra" class on GoogleNet. "w/o Align" and "w/o Fuse" remove the alignment or fusion step, respectively. This complements Table 1 in our paper. Removing align increases variance, while removing fusion inflates all scores, indicating that fusion integrates multi-layer signals to reduce noise and improve consistency.

Concept	**w/o Align**				**w/o Fuse**			
	Mean	Std	CV	RR	Mean	Std	CV	RR
dotted	0.299	0.052	0.175	0.468	0.614	0.297	0.484	1.237
striped	0.844	0.134	0.159	0.426	0.672	0.220	0.327	0.952
zigzagged	0.613	0.139	0.227	0.587	0.627	0.281	0.448	1.197

Table 1. Ablation Study for the "Zebra" class on GoogleNet.

B.2. Hyperparameter Investigation

Table 2 shows TGCAV scores for *striped* on *zebra* using GoogleNet. We vary one setting at a time, fixing others as default (λ_{var}:λ_{cons} = 3:1, λ_{NCE}:λ_{cons} = 1:3, embedding = 4096), which are used in our main paper. We tested the case with no consistency loss ($\lambda_{\text{cons}} = 0$ in both stage). This caused TGCAV scores to collapse to ∼0.5 across layers, discarding meaningful concept signals. This motivated its inclusion in the final design. Embedding size showed minor effect; we selected a higher value (4096) for expressiveness.

Hyperparameter	**Value**	**Mean**	**Std**	**CV**	**RR**
$\lambda_{\text{var}} : \lambda_{\text{cons}}$	1 : 1	0.841	0.172	0.205	0.511
$\lambda_{\text{var}} : \lambda_{\text{cons}}$	1 : 3	0.820	0.192	0.234	0.524
$\lambda_{\text{NCE}} : \lambda_{\text{cons}}$	1 : 1	0.643	0.107	0.166	0.486
$\lambda_{\text{NCE}} : \lambda_{\text{cons}}$	3 : 1	0.510	0.143	0.280	0.506
Embedding dim	1024	0.701	0.042	0.059	0.112
Embedding dim	2048	0.750	0.069	0.092	0.107

Table 2. Hyperparameter Investigation.

B.3. Testing with High-level Concepts on More Datasets

As shown in table 3, we evaluated three high-level concepts on CelebA, StanfordCars, and CUB-200-2011, using 500 images per concept. Our method still reduces inconsistency, showing good generalization.

	TCAV				TGCAV			
Concept (Task)	Mean	Std	CV	RR	Mean	Std	CV	RR
human (groom)	0.333	0.175	0.526	1.740	0.432	0.056	0.130	0.323
car (cab)	0.608	0.176	0.289	0.888	0.692	0.098	0.142	0.103
bird (robin)	0.623	0.132	0.212	0.674	0.639	0.102	0.160	0.496

Table 3. Testing with High-level Concepts on CelebA, StanfordCars, and CUB-200-2011.

B.4. Bar Chart Analysis

Figures 1, 2, and 3 illustrate the distribution of TCAV and TGCAV scores for different concepts across layers.

- **GoogleNet (Figure 1)**: The top row (original TCAV results) shows that the scores fluctuate significantly across layers, making interpretation inconsistent. For instance, the concept "cobwebbed" exhibits high scores in some layers while being nearly absent in others. The bottom row (TGCAV results) demonstrates that our GCAV method stabilizes concept attribution across layers, making it more robust and reliable.
- **MobileNetV2 (Figure 2)**: Similar to GoogleNet, the original TCAV scores exhibit large variations. Notably, the "striped" concept has extremely high TCAV scores in some layers but is nearly absent in others. With TGCAV, the concept scores become more evenly distributed, indicating a more stable interpretation.
- **ResNet50V2 (Figure 3)**: The original TCAV scores fluctuate across layers, with some concepts like "paisley" exhibiting disproportionately high scores in certain layers. The TGCAV results, however, show that concept influence is distributed more consistently across the network, reducing layer dependence.

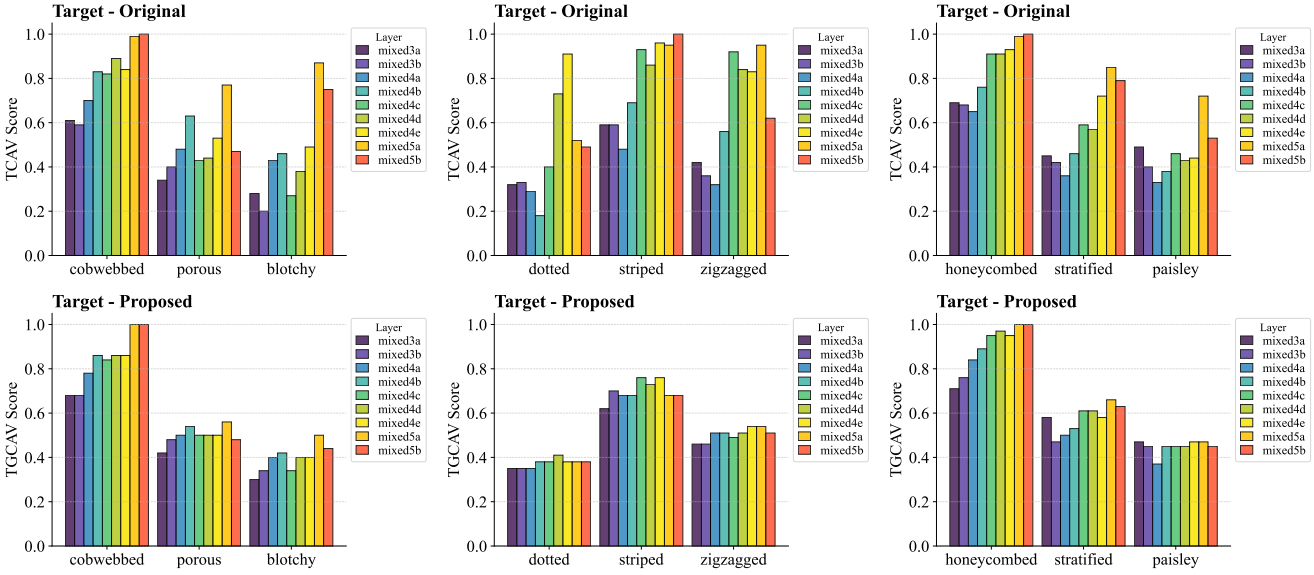

Figure 1. Bar Chart of TCAV scores of GoogleNet.

B.5. Violin Plot Analysis

Figures 4, 5, and 6 present violin plots that visualize the distribution of TCAV and TGCAV scores across layers.

- **GoogleNet (Figure 4)**: The original TCAV scores (top row) display high variance, with some concepts exhibiting a bimodal distribution, indicating instability across layers. For example, the "zigzagged" concept has strong activations in certain layers while being much weaker in others. The TGCAV results (bottom row) show a much more compact distribution, confirming the stabilization effect of our GCAV framework.
- **MobileNetV2 (Figure 5)**: The original TCAV scores exhibit large fluctuations across layers, with some concepts like "honeycombed" showing extreme variations. After applying GCAV, the distributions become significantly more compact,

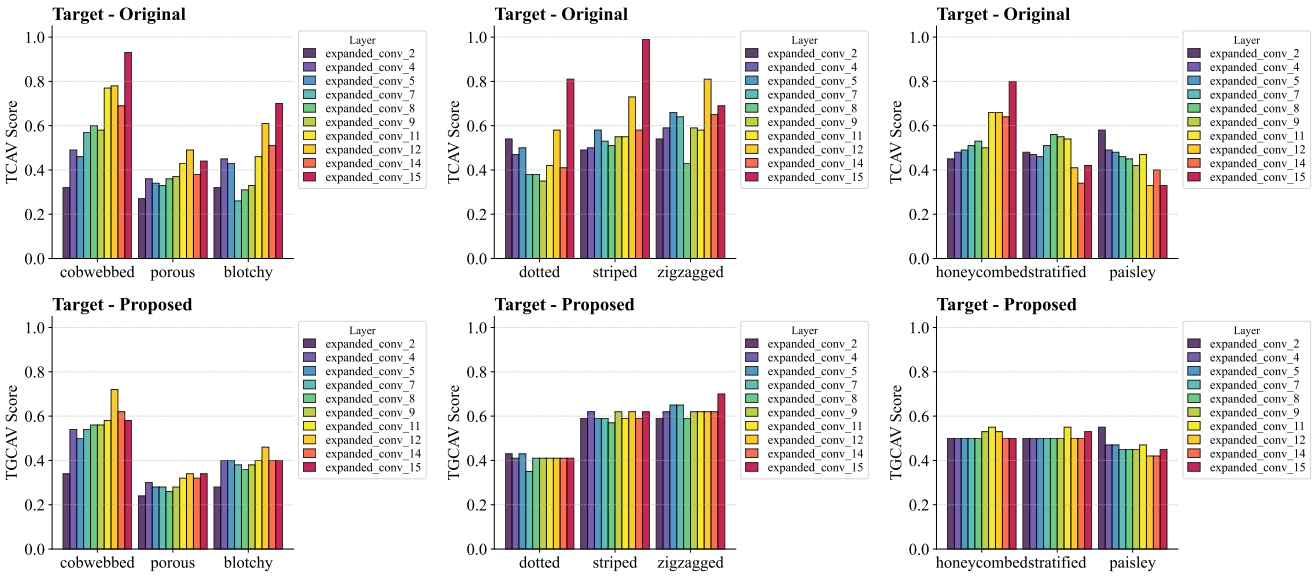

Figure 2. Bar Chart of TCAV scores of MobileNetV2.

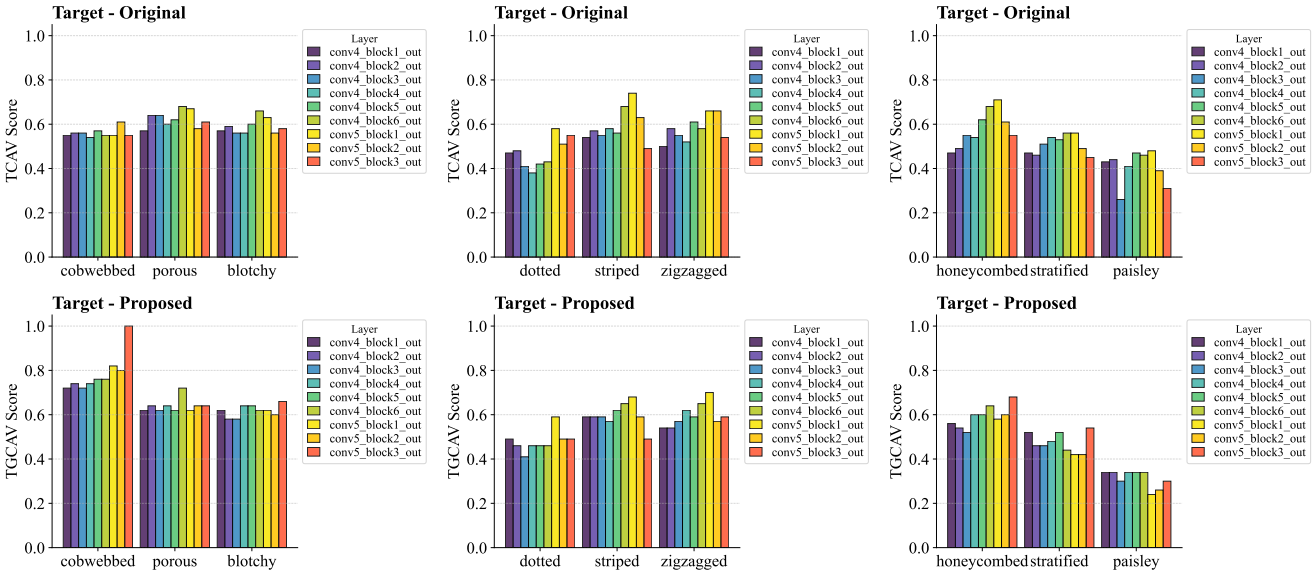

Figure 3. Bar Chart of TCAV scores of ResNet50V2.

ensuring that concept influence is more evenly spread across the layers.

- **ResNet50V2 (Figure 6)**: The original TCAV distributions display significant variability, highlighting the instability of layer-wise concept attribution. The TGCAV results demonstrate that our method successfully reduces variance, making concept attributions more stable across layers.

B.6. Summary

- The **bar charts** show that TGCAV significantly reduces the cross-layer variability of TCAV scores, ensuring more stable concept interpretations.
- The **violin plots** further illustrate that TGCAV leads to more compact distributions, reducing variance and enhancing consistency in concept activations.
- Overall, GCAV improves the reliability of concept-based interpretability by integrating cross-layer information, reducing

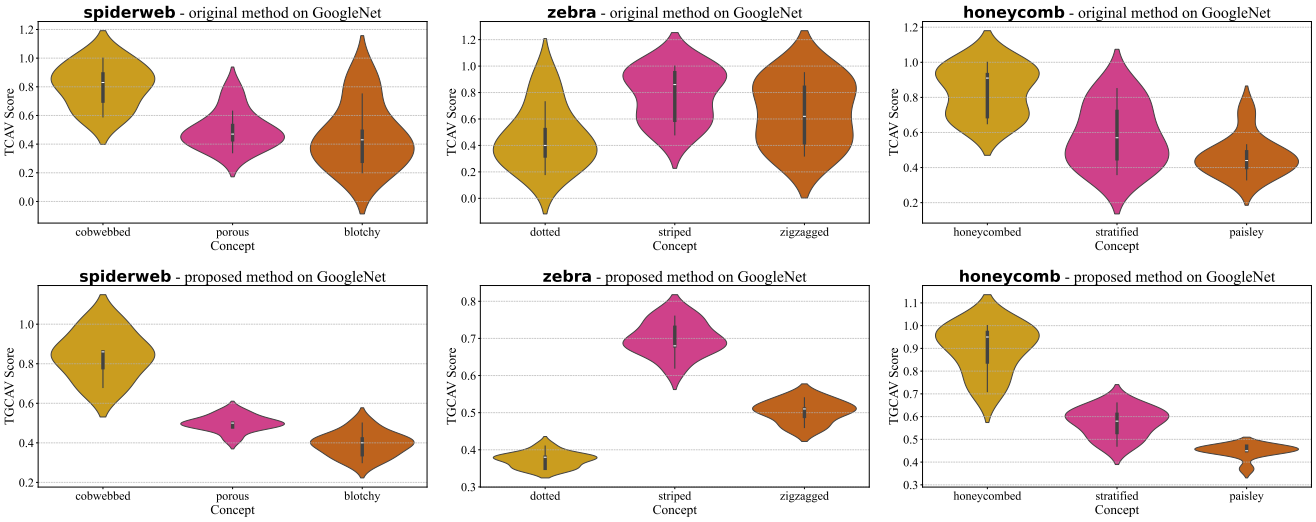

Figure 4. Violin Plots of TCAV scores of GoogleNet.

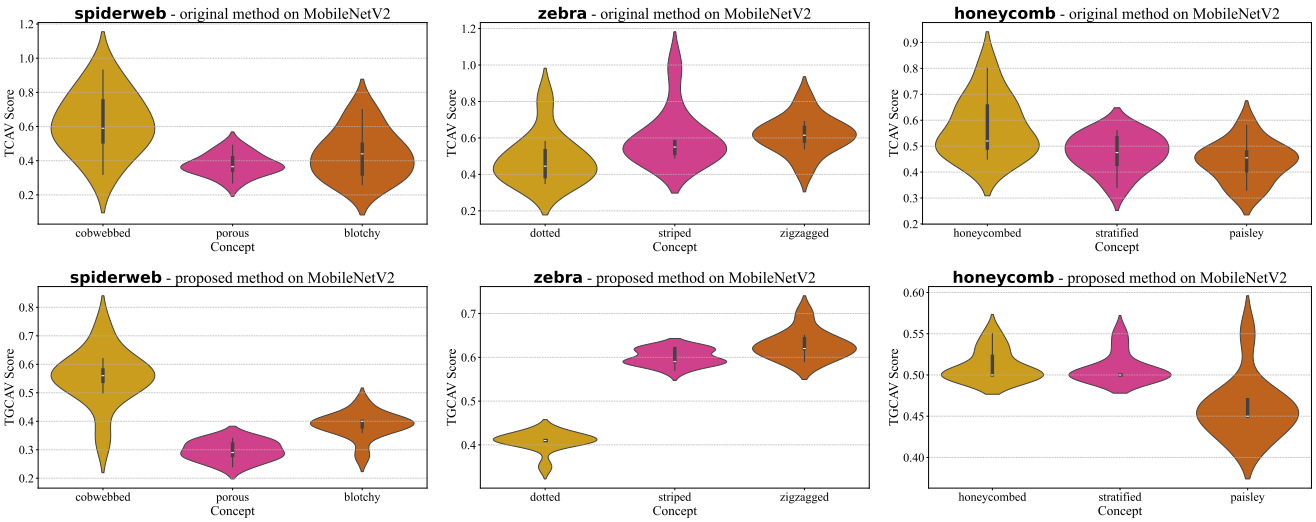

Figure 5. Violin Plots of TCAV scores of MobileNetV2.

layer selection uncertainty, and enhancing robustness against adversarial perturbations.

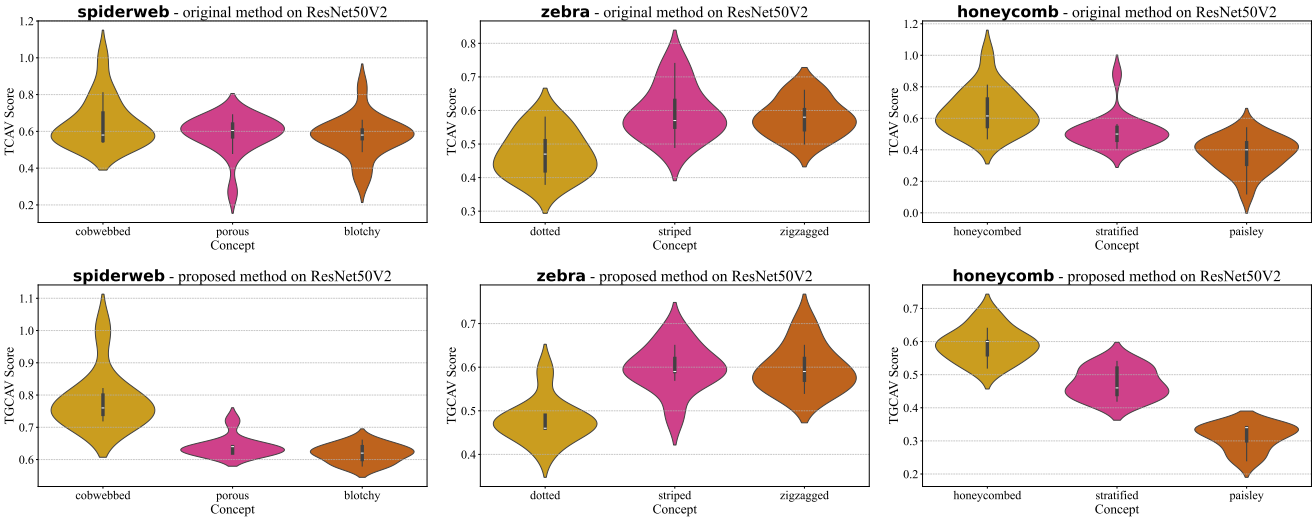

Figure 6. Violin Plots of TCAV scores of ResNet50V2.