Concept Distillation: Leveraging Human-Centered Explanations for Model Improvement Supplementary Material

1 Local loss from RRR

To leverage the local explanation benefits apart from the global ones, we additionally use a local loss from RRR [14] for our experiments over ColorMNIST, DecoyMNIST, and TextureMNIST datasets (indicated by $Ours + L$ in Table 1 and Table 2 in paper). RRR [14] has a reasoning loss, which is added to the original GT loss of a classifier with a factor λ, and is given as:

$$\nabla L_o \left(f_l(\boldsymbol{x}) \right) = \frac{\partial L_o}{\partial \left(f_l(\boldsymbol{x}) \right)} \tag{1}$$

$$\nabla L_p \left(f_l(\boldsymbol{x}) \right) = \frac{\partial L_p}{\partial \left(f_l(\boldsymbol{x}) \right)} \tag{2}$$

Where $A \in \{0, 1\}^{N \times D}$ is the user-provided annotation matrix. It is a binary mask to indicate if dimension D should be irrelevant for the prediction of observation n. λ is a regularization parameter which ensures that "right answers" and "right reasons" terms have similar orders of magnitude. N is the total number of observations and $\hat{y}_{nk}$ is the predicted probability for class k and observation n. RRR penalizes the gradients over the binary mask feature-wise for each input sample and thus is a local XAI method. In the case of debiasing color concept, we take the programmatic rule of single pixel not affecting (single-pixel represents color information) while a group of pixels affecting as demonstrated by [12] for their adaptation of RRR. We additionally experimented by including local interpretability-based losses from CDEP [12], or EG [4] (instead of RRR) for MNIST datasets (ColorMNIST, DecoyMNIST, and TextureMNIST) but found the one in RRR [13] to work best. It is to be noted that local method losses like CDEP [12], RRR [13], or EG [4] require user-given rules. For ColorMNIST and DecoyMNIST, which have the bias of color, they minimize the contribution of individual pixels since the color information is contributed by individual pixels. For the execution of their methods on TextureMNIST, we use the same individual pixel contribution minimization rule. Such rules are not straightforward to extract in case of more complex bias (for example, in BFFHQ), and hence, applying such local XAI-based methods is not feasible.

2 Mapping Module: Case of Ideal Mapping

In theory, if the mapping module has a zero-loss, it could make the distillation case the same as the case without distillation, but this is not observed in our experiments due to two main reasons: *(i)* We use the mapping module to map *only* the conceptual knowledge in CAV and train it only for concept sets and not the training samples. *(ii)* Due to major differences in the perceived notion of concepts in teacher and student networks and due to a simple mapping autoencoder (one upconv and downconv layer) the MSE loss never goes to zero (e.g, in ColorMNIST, it starts from 11 and converges at 5 for DINO teacher to biased student alignment). In our initial experiments, we tried bigger architectures (ResNet18+) and found improved mapping losses but decreased student performances. Mapping Encoder encodes an expert's knowledge into the system via the provided concept sets, quantifies this

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

knowledge as CAV via a generalized teacher model trained on large amount of data, and thus helps in inducing it via distillation into the student model. This brings threefold advantages in our system: expert's intuition, large model's generality, and efficiency of distillation.

3 An experiment: CAV Learning comparison

In ColorMNIST, zeros are always associated with the color *red*. We learn a CAV for the concept of *red* (CAV_{red} separating red patches with other colored patches) in each teacher, mapped teacher and base model (biased initial Student) and measure its cosine similarity (cs) with the respective model representations for concept images of red, red-zeros.

As seen from Tab. 1, the Teacher and Mapped Teacher have cs(red) > cs(red_zeros) < cs(red_non_zeros) which indicate correct concept learning while the Student (Base Model) has cs(red) < cs(red_zeros) < cs(red_non-zeros) that indicates confusion of bias with concept (concept red-zeros confused with concept red). Thus, teacher's CAV and its mapped version capture the intended concept well, while the base model (student) confuses the red-zeros concept with CAV. This small demonstration shows (a) why the teacher is needed for good CAV learning and (b) how well CAV's are transferred to the student via the mapping module. Mapping module does not bias the CAV representation.

Table 1: Cosine Similarity Order of concepts with CAV_{red}

Model	'red'	'red_zeros'	'red_non_zeros'	'non_red_zeros'
Teacher	**0.084**	**-0.013**	-0.009	0.005
Mapped Teacher	0.287	-0.044	0.014	0.024
Base Model	-0.023	-0.019	-0.013	0.000

4 Fixing vs Varying CAVs in Experiments

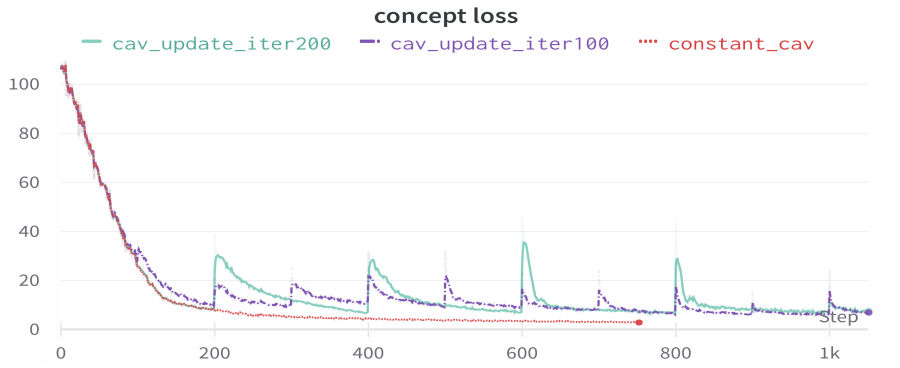

Figure 1: Loss Curves in varying cav update iterations: frequent pattern observed, convergence quickly

In our experimentation reported in main paper, we fix CAVs but we experimented with updating CAVs in the student after every few training iterations (50, 100, 200, etc.). Specifically, we experimented in the following settings and report the concept loss L_c curves with iterations in Fig. 1.

- cav_update_iter200: CAV updates every 200 iterations.
- cav_update_iter100: CAV updates every 100 iterations.
- constant_cav: Fixed CAV throughout training.

As seen in Fig. 1, there is a recurring pattern with concept loss L_C increasing on CAV update (due to an abrupt change in objectives), followed by a subsequent decrease due to optimization. However, L_C

remains lower than the initial value the model started with (from 105.98 at iteration 0, loss dropped to 9.51 at iteration 199 (before the CAV update) but jumps to 20.61 following the CAV update), underscoring the efficacy of our approach. Similar patterns were consistently observed when updating CAVs in varying numbers of iterations. This trend persisted across diverse datasets during training as well.

The given graph is shown for experimentation over ColorMNIST, but we find the same recurring patterns across all other datasets. Additionally, the best validation accuracy (and also corresponding test accuracy) values for all the settings mentioned above (whether fixed or varying CAVs) are the same (within < 0.3% accuracy changes amongst the settings). The graph also shows that our design choice of keeping CAVs fixed is good practically as the loss quickly converges to a lower value.

5 Additional Ablations

Table 2: Impact of different variants of CAV sensitivity calculation gradient ∇X in proposed loss Eq. 2 on the final results

X	logit	L_o	L_p, fixed proto	L_p, varying prototypes (Ours)
Accuracy %	25.55	30.94	40.02	41.83

We evaluate the impact of different components of our framework apart from the ones already mentioned in the paper.

We experimented with different ways of calculating sensitivity used for L_c in Eq. 2 by replacing ∇L_o with ∇X where X is taken as *(i)* last layer outputs or logits; *(ii)* last layer loss L_o and *(iii)* intermediate layer prototype based loss L_p in two settings: fixed prototypes where prototypes are kept fixed as initial ($\alpha = 0$) vs prototypes are varied with α. Settings *(i)* and *(ii)* are essentially final layer sensitivity calculation according to the original implementation by Kim et al. [6] while Setting *(iii)* is our proposed intermediate layer sensitivity using prototypes. We also show results when *(a)* KNN k in the prototype calculation is varied and found k=7 to work best Fig. 2 *(b)* number of images (#imgs) in the concept set are varied and we observe a peak in #imgs = 150 which is chosen as the KNN k and #imgs in our experiments.

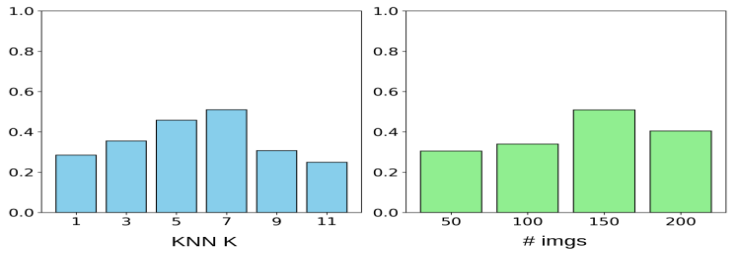

Figure 2: Impact of varying the concept set size and number of means on 100% biased ColorMNIST.

6 Design Choices and Implementation Details

Choosing the Student Layer for Concept Distillation: CAVs can be calculated for any model layer. Which student layers should be used? We show results only for the last convolution layer of the student model in the paper, but theoretically, our framework can be extended to any number of layers at any depth. Our design choice is based on the fact that the deeper layers of the model encode complex higher-order class-level features while the shallower layers encode low-level features. The last convolution layer represents more abstract features, which are easily representable for humans in the form of concepts rather than low-level features in other layers. For the same reasons, [1] too use the last convolution layers for conceptual explanation generation.

Teacher Selection: For a teacher, we experimented with various model architectures (including the biased teacher ones shown in Sec. 4 Ablations) and chose a pre-trained DINO transformer [2] for

the main reason of scalability. DINO has been proven to work well for a variety of tasks [15, 17]. A large model knows a variety of concepts and can be used as a teacher for various tasks, as shown in our experiments. We use the same DINO teacher on very different classification datasets like biased MNIST (ColorMNIST, DecoyMNIST, and TextureMNIST) and BFFHQ, as well as over a completely different problem of IID.

Among the DINO variants, we found ViT-B/8 (85M parameters) to perform the best, aiding student to get 50.93% accuracy on ColorMNIST while ViT-S/8 (21M parameters) aced 39% student accuracy. We thus picked DINO ViT-B/8 for all our experiments. We used the code implementation of the DINO feature extractor by Tschernezki et al. [16] and loaded the checkpoints for DINO variants from [2]. The DINO ViT-B/8 gives 768-dimensional feature images, further reduced to 64 using PCA.

Apart from biased small teacher experiments in Tab. 6 (as shown in the paper) where we vary the bias in teacher from 5% to 100%, we also experimented when the small teacher network is (pre)trained on 0% bias (simply MNIST dataset [8]). We found the student to achieve an accuracy of 23.6% with this teacher network. This accuracy is lesser than that in biased teacher due to the fact that the teacher trained on the MNIST dataset having grayscale images has never seen the concept of color. Henceforth as mentioned before, it is important for teacher to have the knowledge of concepts for our proposed concept distillation to work well.

CAV learning: For CAV learning, we use a logistic regression implemented by a single perceptron layer with sigmoid activation. We train it to distinguish between model activations of concept set (C) and its negative counterpart (C') in layer l using a cross-entropy loss for binary classification. This is theoretically the same but implementation-wise slightly different from [6], who also use a logistic regression but from sklearn [11]. We get the same results from either of them, though our perceptron-based implementation is slightly faster in terms of computation.

Mapping Module details: For the mapping module, we choose a pair of one down-convolutional and up-convolutional layers as Encoder and decoder (depending on students and teacher's dimensions, it is determined whether the Encoder is up or down-convolutional and vice versa). In our experiments, we train the autoencoder for a maximum of five epochs and select the Encoder from the best of the first five epochs as our activation space mapping module M. We also tried with other deeper autoencoder architectures in our initial experiments but found the above simple one to give good results while being computationally cheapest. Due to the simple architecture (logistic regression or single up-down convolutions), both our CAV learning and Mapping module training are very lightweight (< 120K parameters, < 10MB and <1MB) and train within a minute for 10-15 concept sets having a number of images between 50-200 on a single 12GB Nvidia 1080 Ti GPU.

Taking only Robust CAVs for Distillation [6] use t-testing with concept vs multiple random samples to filter robust CAVs for TCAV score estimation. Similar to them, we employed t-testing initially by taking concept vs multiple random samples and selecting only the significant CAVs. This proved to be too expensive computationally during training, especially during frequent CAV updates. Currently, we have a simple filter on CAV classification accuracy > 0.7 to select only the good CAVs (i.e., CAVs that can differentiate concept vs random). The concept loss corresponding to all such valid CAVs is then averaged before backpropagating. This design simplification was empirically verified and found to work equivalently to [6] t-testing).

6.1 Debiasing in classification details

For the student network in classification debiasing applications, we use two convolution layers followed by two fully connected layers as done by [12] [9] for all three biased MNIST experiments. We use Resnet18 (no pre-training) for BFFHQ student architecture as done by [9] and apply our method over the "layer4.1.conv1" layer. We use Adam [7] optimizer with a learning rate of 10e-4, $0.9 \leq \beta \leq 0.999$, and $ep = 1e-08$ with a weight decay of 0 (all default pytorch values except learning late). We use a batch size of 32 for MNIST experiments and 64 for BFFHQ experiments. For the mapping module, we use one up-convolution and one down-convolution layer for encoders and decoders. We train the autoencoders with an L2 loss. For training the MNIST and BFFHQ models, we use the cross-entropy loss as the Ground Truth loss (L_o). Our concept loss L_C is weighted by a parameter λ varied from 0.01 to 10e5 in our experiments. We found it to work best for values close to 20 in our experimentation. Other parameter values that we found to work best are number of clusters

Figure 3: pixel hard MNIST test set

in K-Means $k = 7$ and prototypes updation weight $\alpha = 0.3$. Our student model converges within 2-3 epochs of training with a training time of less than 1.5-2 hours for MNIST datasets and within 4 hours for the BFFHQ dataset (on one Nvidia 1080 Ti GPU).

6.2 MNIST datasets

For MNIST datasets (ColorMNIST and DecoyMNIST), we use the splits by Rieger et al. [12]. For the creation of TextureMNIST, we use the above-obtained splits of digits and replace the colors with textures from DTD [3] (all colors removed and rather texture bias added). We use random flat-colored patches as the concept "color," random textures as the "textures" concept, and gray-colored patches as the "gray" concept set. For the negative concept set, we create randomly shaped white blobs in black backgrounds. ColorMNIST and TextureMNIST datasets have test sets comprising flipped colors and random textures digits. To check the generalization of our model, we tested the ColorMNIST and TextureMNIST trained models on other test sets having random colors and textures (Tab. 4) . Also, we created our Pixel Hard MNIST test set (Fig. 3), which has color in each pixel in digits randomized in contrast to one color in the entire digit in others(ColorMNIST and TextureMNIST).

6.3 BFFHQ Details

We take the 48 images each for young men and old women (bias conflicting samples as given by Lee et al. [9]) as our concept sets for class-specific training (separate concepts for each class). For the negative concept set of class-specific training, we take mixed images of both young and old women for women concept and young and old men for men concept. We use the same train-test-validation split as done by [9].

6.4 IID details

For our IID students, we use the architecture and training datasets from [10]. For L_o, we adopt the same losses as Li and Snavely [10], which consist of an IIW loss, GT Reconstruction loss, and a SAW loss. We train the model for 5-10 epochs, saving the best checkpoint (on validation). We show the IID concept sets example in Fig. 4. For IID concept sets, we follow [5]. Their albedo variations Δa concept set consists of scenes in fixed illumination and camera viewpoint with varying albedo (or intrinsic color). For the illumination variation Δi concept set, they fix the albedo and viewpoint in the scene while varying the illumination direction. They have multiple such scenes consisting of both single and multiple objects rendered in the blender. We use 100 images per scene for Δa and 44 for Δi, similar to them. For more details, please refer [5]. Our model trains within 16-20 hours for IID on two 12GB Nvidia 1080 Ti GPUs.

7 Additional results

We show the additional IID results in Fig. 5. It can be seen that our method is able to remove the illumination information from it and flatten the predictions of R ($\hat{R}$) as well.

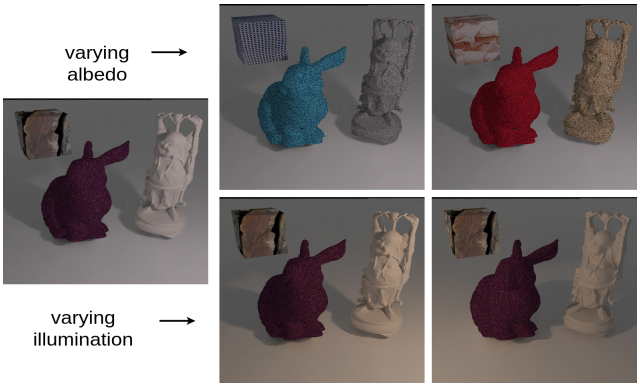

Figure 4: Concept sets used for IID experiments

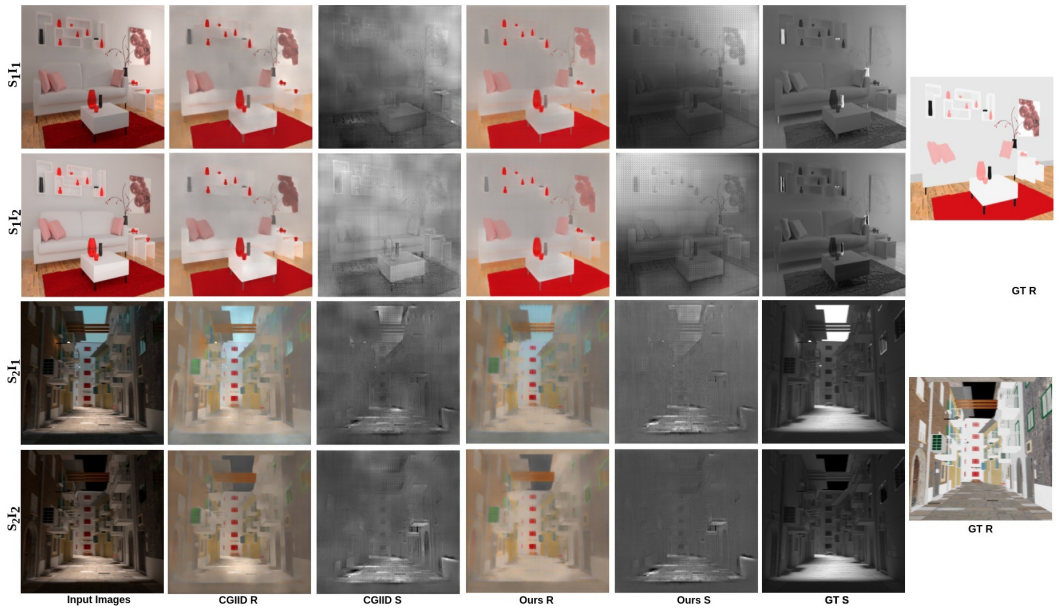

Figure 5: Qualitative IID results: Our method is able to make the $\hat{R}$ less sensitive to illumination (thereby removing the concept of illumination from $\hat{R}$ and during this $\hat{R}$ predictions become flatter without specifically introducing the flatness prior suggesting disentanglement of R-S is a better way to improve IID. Also, the illumination information removed from $\hat{R}$ is introduced in $\hat{S}$.

References

[1] A. Akula, S. Wang, and S.-C. Zhu. Cocox: Generating conceptual and counterfactual explanations via fault-lines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2594–2601, 2020.

[2] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[3] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[4] G. Erion, J. D. Janizek, P. Sturmfels, S. M. Lundberg, and S.-I. Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature machine intelligence*, 3 (7):620–631, 2021.

[5] A. Gupta, S. Saini, and P. J. Narayanan. Interpreting intrinsic image decomposition using concept activations. In *Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image*

Processing, ICVGIP '22, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450398220. doi: 10.1145/3571600.3571603.

[6] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[9] J. Lee, E. Kim, J. Lee, J. Lee, and J. Choo. Learning debiased representation via disentangled feature augmentation. *Advances in Neural Information Processing Systems*, 34:25123–25133, 2021.

[10] Z. Li and N. Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *European Conference on Computer Vision (ECCV)*, 2018.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] L. Rieger, C. Singh, W. Murdoch, and B. Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International conference on machine learning*, pages 8116–8126. PMLR, 2020.

[13] A. S. Ross, M. C. Hughes, and F. Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

[14] P. Schramowski, W. Stammer, S. Teso, A. Brugger, F. Herbert, X. Shao, H.-G. Luigs, A.-K. Mahlein, and K. Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486, 2020.

[15] V. Tschernezki, I. Laina, D. Larlus, and A. Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. *arXiv preprint arXiv:2209.03494*, 2022.

[16] V. Tschernezki, I. Laina, D. Larlus, and A. Vedaldi. Neural feature fusion fields: 3D distillation of self-supervised 2D image representations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022.

[17] X. Wanyan, S. Seneviratne, S. Shen, and M. Kirley. Dino-mc: Self-supervised contrastive learning for remote sensing imagery with multi-sized local crops. *arXiv preprint arXiv:2303.06670*, 2023.