ProtoVAE: A Trustworthy Self-Explainable Prototypical Variational Model (Supplementary Material)

S1 Introduction

In this supplementary document we provide additional details for the proposed ProtoVAE. Specifically, we provide the derivation to illustrate that ProtoVAE learns a mixture of VAEs, followed by additional dataset, training and implementation details. We further provide additional results including ablation studies for the loss terms, predictive performance using ResNet-18 [1] as encoder, further comparisons with FLINT [2], and demonstrate ProtoVAE's ability to capture the diversity of the data using additional qualitative results. Finally, we discuss the potential negative societal impacts of the proposed method.

S2 Mixture of VAE Derivation

In this section, we provide the derivations leading to Eq. 6 in the main paper. First, let's recall, that training a VAE aims to maximize the evidence lower bound (ELBO), which, assuming the prior on $\boldsymbol{z}$ is a standard Gaussian distribution, can be stated as follows:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p_f(\boldsymbol{z}|\boldsymbol{x})}\big(\log p_g(\boldsymbol{x}|\boldsymbol{z})\big) - D_{\text{KL}}\big(p_f(\boldsymbol{z}|\boldsymbol{x})||\mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)\big), \tag{1}$$

where f and g designate the encoder and decoder, respectively, with $p_f(\boldsymbol{z}|\boldsymbol{x})$ being the approximate posterior distribution and $p_g(\boldsymbol{x}|\boldsymbol{z})$ the likelihood distribution. Assuming that $\boldsymbol{x}|\boldsymbol{z}$ follows a Gaussian distribution with covariance $\frac{1}{2}\boldsymbol{I}_p$, the first term of the $\mathcal{L}_{\text{ELBO}}$ simplifies as follows:

$$\mathbb{E}_{p_f(\boldsymbol{z}|\boldsymbol{x})}\big(\log p_g(\boldsymbol{x}|\boldsymbol{z})\big) = -||\boldsymbol{x} - \bar{\boldsymbol{x}}||^2 + \text{Constants}. \tag{2}$$

Our proposed ProtoVAE model learns a mixture of VAEs sharing the same network each having a Gaussian prior centered on one of the prototypes. More precisely, we consider K mixtures of VAEs each with M components, i.e., one mixture per class and one component per prototype. Each mixture is trained using the training data of one class $k \in 1, \ldots, K$ and the mixture weights are given by the corresponding similarities of all M class prototypes, which are normalized as follows:

$$\tilde{\boldsymbol{s}}_i(k,j) = \frac{\boldsymbol{s}_i(k,j)}{\sum_{l=1}^{M} \boldsymbol{s}_i(k,l)}.$$

Hence, the loss function becomes:

$$\begin{aligned}
\mathcal{L}_{\text{VAE}} =& \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{M} -\boldsymbol{y}_i(k)\tilde{\boldsymbol{s}}_i(k,j)\mathcal{L}_{\text{ELBO}}(i,k,j). \\
=& \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{M} \boldsymbol{y}_i(k)\tilde{\boldsymbol{s}}_i(k,j)\Big(||\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i||^2 - D_{\text{KL}}\big(\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)||\mathcal{N}(\boldsymbol{\phi}_{kj}, \mathbf{I}_d)\big)\Big).
\end{aligned} \tag{3}$$

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Since $\boldsymbol{y}_i$ is a one-hot encoded vector, the reconstruction term simplifies to:

$$\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{M}\boldsymbol{y}_i(k)\tilde{\boldsymbol{s}}_i(k,j)||\boldsymbol{x}_i-\bar{\boldsymbol{x}}_i||^2=\sum_{i=1}^{N}||\boldsymbol{x}_i-\bar{\boldsymbol{x}}_i||^2\sum_{j=1}^{M}\tilde{\boldsymbol{s}}_i(\boldsymbol{y}_i(k),j)=\sum_{i=1}^{N}||\boldsymbol{x}_i-\bar{\boldsymbol{x}}_i||^2. \quad (4)$$

Now, injecting this simplification into Eq. 3 yields Eq. 6 of the main paper:

$$\mathcal{L}_{\text{VAE}}=\frac{1}{N}\sum_{i=1}^{N}||\boldsymbol{x}_i-\bar{\boldsymbol{x}}_i||^2-\sum_{k=1}^{K}\sum_{j=1}^{M}\boldsymbol{y}_i(k)\tilde{\boldsymbol{s}}_i(k,j)D_{\text{KL}}\big(\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\sigma}_i)||\mathcal{N}(\boldsymbol{\phi}_{kj},\mathbf{I}_d)\big). \quad (5)$$

S3 Dataset details

The additional details, including the number of classes, number of training and testing samples, image sizes and license for the datasets are provided in Table 1. For MNIST, fMNIST, CIFAR-10, and SVHN, the default splits are used for training and testing. For QuickDraw, we use the subset created by [2][1], which consists of 10000 random images per class from the following 10 classes: 'Ant', 'Apple', 'Banana', 'Carrot', 'Cat', 'Cow', 'Dog', 'Frog', 'Grapes', 'Lion'. Of the 10000 images, 8000 are used for training and 2000 for testing. As pre-processing, all datasets were normalized to lie in the range [-1,1]. No data augmentation was performed for MNIST, fMNIST, QuickDraw and SVHN. For CIFAR-10, we first used zero-padding of size 2 and afterwards cropped images of size 32x32, which were then further used for training, following [2]. We further apply color jittering, where brightness, contrast, saturation, and hue of an image are changed randomly, as well as random horizontal flipping with a probability of 0.5.

Table 1: Additional details for the open-source datasets used in this work.

	MNIST	**Fashion-MNIST (fMNIST)**	**QuickDraw (subset [2])**	**CIFAR-10**	**SVHN**
No. of classes	10	10	10	10	10
Number of samples (Training)	60,000	60,000	80,000	60,000	73,257
Number of samples (Testing)	10,000	10,000	20,000	10,000	26,032
Image size	28x28	28x28	28x28	32x32	32x32
License	CC BY-SA 3.0	MIT	CC BY 4.0	MIT	CC0 1.0

S4 Training and Implementation

All the experiments are conducted using PyTorch [3] with a NVIDIA GeForce GTX 1080 Ti GPU. The detailed network architectures for all datasets are shown in Figure 1. The hyperparameter settings used for different datasets are described in Table 2 for ProtoVAE, for the Black-box encoder in Table 3, and for ProtoPNet in Table 4. For ProtoPNet, the optimal values for learning rates and coefficients for the loss terms are used, as reported in [4].

Table 2: Hyperparameter and training settings for the different datasets for ProtoVAE.

	MNIST	**fMNIST**	**CIFAR-10**	**QuickDraw**	**SVHN**
No. of prototypes per class	5	10	10	10	5
Size of feature vector (z)	256	256	512	512	512
No. of epochs	10	10	36	10	10
Batch size	128	128	128	128	64
Learning rate	0.001	0.001	0.001	0.001	0.001

S5 Source Code

Our source code is available at https://github.com/SrishtiGautam/ProtoVAE.

[1] `https://github.com/jayneelparekh/FLINT`

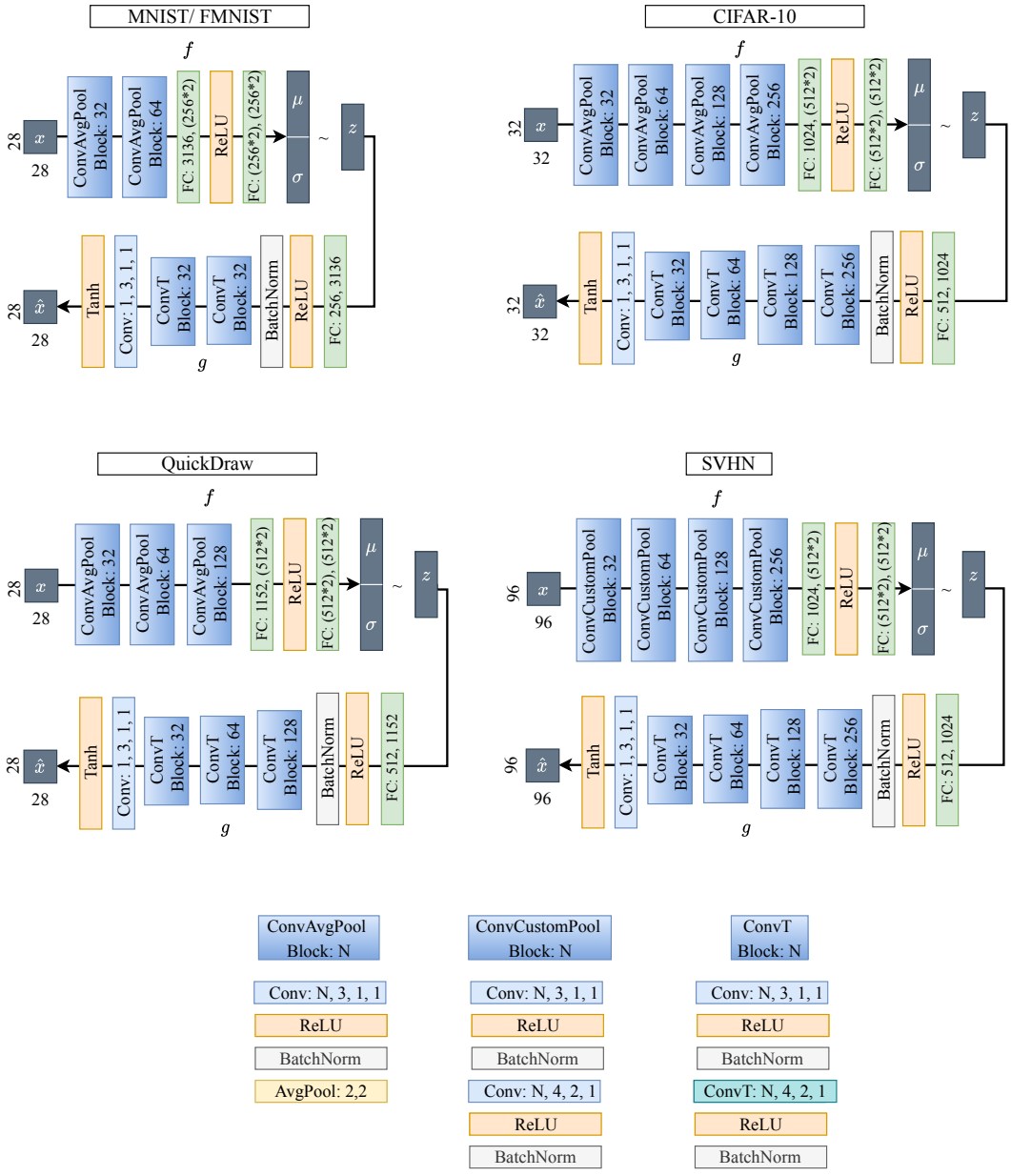


Figure 1: Visualization of the architectures used for the different datasets. f is the encoder and g the decoder. Conv: i,j,k,l represents the convolutional layer with number of output channels i, kernel size j, stride k and padding l. FC: i,j represents the fully connected layer with i input neurons and j output neurons. ConvT represents a transposed convolutional layer. AvgPool: i,j represents average pooling layer with kernel size i and stride j.

Table 3: Hyperparameter and training settings for the different datasets for Black-box encoder.

	MNIST	**fMNIST**	**CIFAR-10**	**QuickDraw**	**SVHN**
No. of epochs	10	10	40	4	7
Batch size	128	128	128	128	64
Learning rate	0.001	0.001	0.001	0.001	0.001

Table 4: Hyperparameter and training settings for the different datasets for ProtoPNet.

	MNIST	**fMNIST**	**CIFAR-10**	**QuickDraw**	**SVHN**
Prototype shape	(50,256,1,1)	(100,256,1,1)	(100,512,1,1)	(100,512,1,1)	(50,512,1,1)
No. of epochs	31	31	21	51	51
Batch size	128	128	128	128	64

S6 Additional Results

In this section, we provide additional quantitative as well as qualitative results for ProtoVAE.

S6.1 Ablation Study

In the following, we conduct an ablation study for the terms in Eq. 2 in the main paper. Specifically, we study the influence of the orthonormalization, ($\mathcal{L}_{\text{orth}}$) and the KL divergences ($\mathcal{L}_{\text{KL}}$), where:

$$\mathcal{L}_{\text{orth}} = \sum_{k=1}^{K} ||\bar{\mathbf{\Phi}}_k^T \bar{\mathbf{\Phi}}_k - \boldsymbol{I}_M||_F^2, \tag{6}$$

$$\mathcal{L}_{\text{KL}} = \sum_{k=1}^{K} \sum_{j=1}^{M} \boldsymbol{y}_i(k) \frac{\boldsymbol{s}_i(k,j)}{\sum_{l=1}^{M} \boldsymbol{s}_i(k,l)} D_{\text{KL}}\big(\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)||\mathcal{N}(\boldsymbol{\phi}_{kj}, \mathbf{I}_d)\big) \tag{7}$$

$\mathcal{L}_{\text{KL}}$ is the second term in the $\mathcal{L}_{\text{VAE}}$ loss (see Eq. 6 in the main paper). The predictive performance for the MNIST dataset in the presence of: (i) the full loss ($\mathcal{L}_{\text{ProtoVAE}}$), i.e, with the presence of both the above mentioned terms, (ii) without $\mathcal{L}_{\text{orth}}$, i.e, $\mathcal{L}_{\text{ProtoVAE}} - \mathcal{L}_{\text{orth}}$, (iii) without $\mathcal{L}_{\text{KL}}$, i.e, $\mathcal{L}_{\text{ProtoVAE}} - \mathcal{L}_{\text{KL}}$, and (iv) without both, i.e, $\mathcal{L}_{\text{ProtoVAE}} - \mathcal{L}_{\text{orth}} - \mathcal{L}_{\text{KL}}$ is provided in Table 5. Further, the prototypes learned in the presence of full loss and in the absence of $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{KL}}$ are shown in Fig. 2. The absence of the orthogonality term leads to a decrease in accuracy due to a collapse of prototypes to the center of the class. The absence of the KL term leads to a considerable drop in accuracy, as well as poor reconstructions of the prototypes.

Table 5: Ablation study: Prediction accuracy (in %) for MNIST dataset. Mean and standard deviation of 4 runs are provided.

	$\mathcal{L}_{\text{ProtoVAE}}$	*without* $\mathcal{L}_{\text{orth}}$	*without* $\mathcal{L}_{\text{KL}}$	*without both*
MNIST	**99.4±0.1**	99.2±0.1	58.8±10.7	98.9±0.1

S6.2 ProtoVAE with ResNet-18 encoder

In this section we provide the results for all 5 datasets using an ImageNet [5] pretrained ResNet-18 [1] as encoder f. The ResNet-18 layers are followed by add-on layers (the fully connected layers from the encoders in Fig. 1) to provide the same dimensional embeddings as the prior architectures. The decoders follow the same architecture as described in Fig. 1. The hyperparameter and training settings are further provided in Table 6. The predictive performance for all the datasets, as shown in Table 7 demonstrates the effectiveness of ProtoVAE even with a more complex architecture of ResNet-18, thus confirming its wide applicability.

Additionally, note that ProtoVAE can easily be modified for domains where a larger effect on different terms is desired. For example, if more clear global explanations are required for safety critical applications, such as healthcare, the reconstruction loss term can be assigned a higher weight. An example of prototypes learned for the SVHN dataset with ResNet-18 as encoder and 10 as the weight for the reconstruction term are shown in Fig. 3.

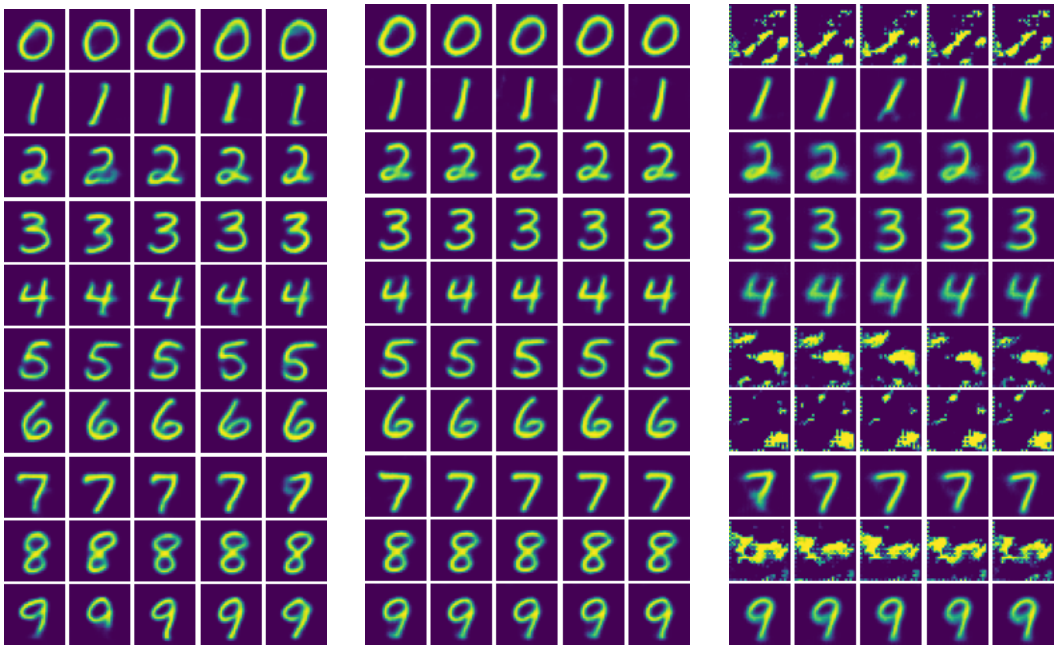


Figure 2: MNIST prototypes with full $\mathcal{L}_{\text{ProtoVAE}}$ loss (left), without $\mathcal{L}_{\text{orth}}$ (middle) and without $\mathcal{L}_{\text{KL}}$ (right). The absence of $\mathcal{L}_{\text{orth}}$ leads to a collapse of prototypes to the center of the class. The absence of $\mathcal{L}_{\text{KL}}$ leads to unreliable reconstructions due to the non-regularized prototype space.

Table 6: Hyperparameter and training settings for the different datasets for ProtoVAE with ResNet-18 as backbone.

	MNIST	**fMNIST**	**CIFAR-10**	**QuickDraw**	**SVHN**
No. of prototypes per class	5	10	10	10	5
Size of feature vector (z)	256	256	512	512	512
No. of epochs	15	15	36	15	15
Batch size	128	128	128	128	64
Learning rate	0.001	0.001	0.001	0.001	0.001

Table 7: Accuracy (in%) for ProtoVAE with ResNet-18 as backbone. The numbers provided are the mean and standard deviation of 4 runs.

	MNIST	**fMNIST**	**CIFAR-10**	**QuickDraw**	**SVHN**
ProtoVAE (ResNet-18)	99.4±0.0	91.9±0.1	80.5±0.4	86.3±0.3	92.2±0.2

S6.3 Comparison with FLINT

We compare the predictive performance of ProtoVAE to the one that FLINT (FLINT-g) and SENN obtain using FLINT's more complex encoders as reported in [2] in Table 8. Even when using FLINT's ResNet-18 [1] based architectures as backbones for QuickDraw and CIFAR-10, ProtoVAE still surpasses their accuracy while only using small encoders.

S6.4 ProtoPNet vs ProtoVAE: Diversity of prototypes

To demonstrate the importance of intra-class *diversity*, in this section we show a few prototypes for the QuickDraw dataset learned by ProtoPNet and ProtoVAE for class 'Lion' (Fig. 4), class 'Banana' (Fig. 5) and class 'Carrot' (Fig. 6). It is observed that ProtoPNet uses 'artifact' training images as prototypes very frequently for several classes, therefore failing to achieve competitive accuracy for

Figure 3: Prototypes learned for SVHN for ResNet-18 as encoder with weight for reconstruction loss as 10. The accuracy achieved by this model is 91.2%.

Table 8: Accuracy comparison (in %) with FLINT and SENN, where their results are obtained from [2]. Note that in this case FLINT and SENN uses more complex encoders, while ProtoVAE results are obtained only using the small encoder. The numbers provided are the mean and standard deviation of 4 runs.

	MNIST	fMNIST	CIFAR-10	QuickDraw
ProtoVAE	**99.4±0.1**	**91.9±0.2**	**84.6±0.1**	**87.5±0.1**
FLINT [2]	98.3±0.2	86.8±0.4	84.0±0.4	85.4±0.1
SENN [6]	98.4±0.1	84.2±0.3	77.8±0.7	85.5±0.4

this dataset. ProtoVAE on the other hand learns diverse prototypes, which are closer to the 'true' mean of the subset of classes, therefore acting as improved class representatives.

S6.5 ProtoPNet vs ProtoVAE: Patch and Image-based explanations

ProtoPNet provides patch-based explanations, where a patch from the training image is used by the network for classification. While, on the other hand, ProtoVAE uses full images as prototypes, which are used for classification. We provide a comparison of patch-based vs image-based explanations provided by these methods in Fig. 7 and observe that ProtoPNet loses contextual information due to their patch-based prototypes. This highlights their dependency on the original images where the prototype patches come from to be interpretable. ProtoVAE instead uses full image embeddings as

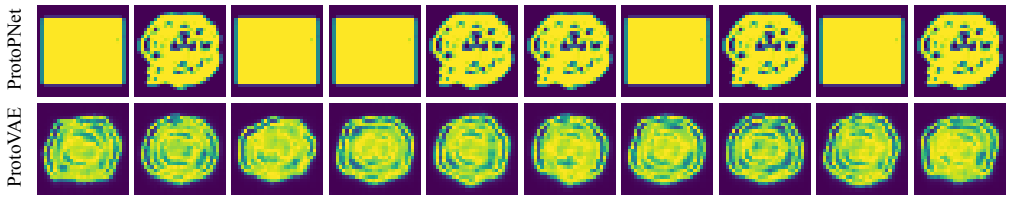


Figure 4: Prototypes learned by 'Lion' class of QuickDraw dataset for ProtoPNet (top) and ProtoVAE (bottom).

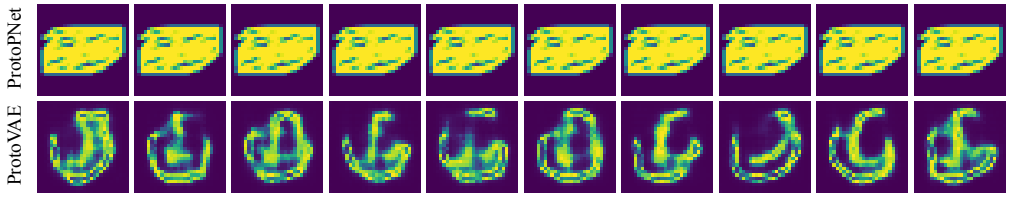


Figure 5: Prototypes learned by 'Banana' class of QuickDraw dataset for ProtoPNet (top) and ProtoVAE (bottom).

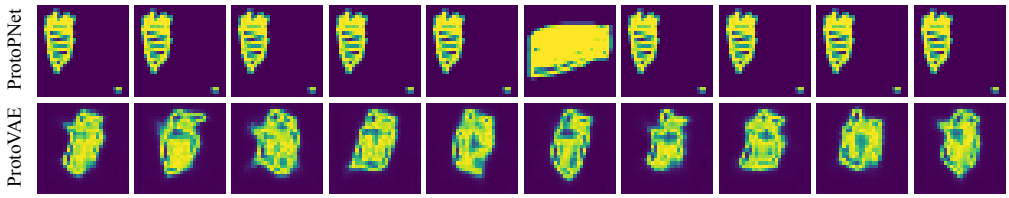


Figure 6: Prototypes learned by 'Carrot' class of QuickDraw dataset for ProtoPNet (top) and ProtoVAE (bottom).

prototypes, which efficiently represent the 'true' mean of class subsets and are thus fully interpretable on their own.

S6.6 ProtoPNet vs ProtoVAE: Visualization of prototypes

We compare the prototype visualization techniques used by ProtoPNet and ProtoVAE. While ProtoPNet imitates transparency by projecting prototypes to the closest training images, ProtoVAE has the capability to visualize the learned prototypes with the help of the in-built decoder architecture. We show the prototypes learned by ProtoPNet and ProtoVAE for the SVHN dataset in Fig 8. SVHN consists of many training samples with multiple digits in each image. Since ProtoPNet limits the prototypes to be represented by training samples, its prototypes images also capture images with multiple digits. ProtoVAE however allows more flexibility and interestingly learns prototypes that only capture the correct class digit.

S6.7 Enhanced interpretability by local explainability maps

While the quality of reconstruction is limited by the simple VAE, the robust local explanability maps produced by ProtoVAE offer an additional way to explain the predictions or the similarity scores in a situation where the visualizations of the prototypes are not very informative. In Figure 9, we show prototypical explanations generated using LRP for 4 test images from CIFAR-10 dataset. For each test image, the LRP maps for 2 prototypes from different classes are shown to analyse the area of interest in the test image by different class prototypes. As can be observed, the first test image

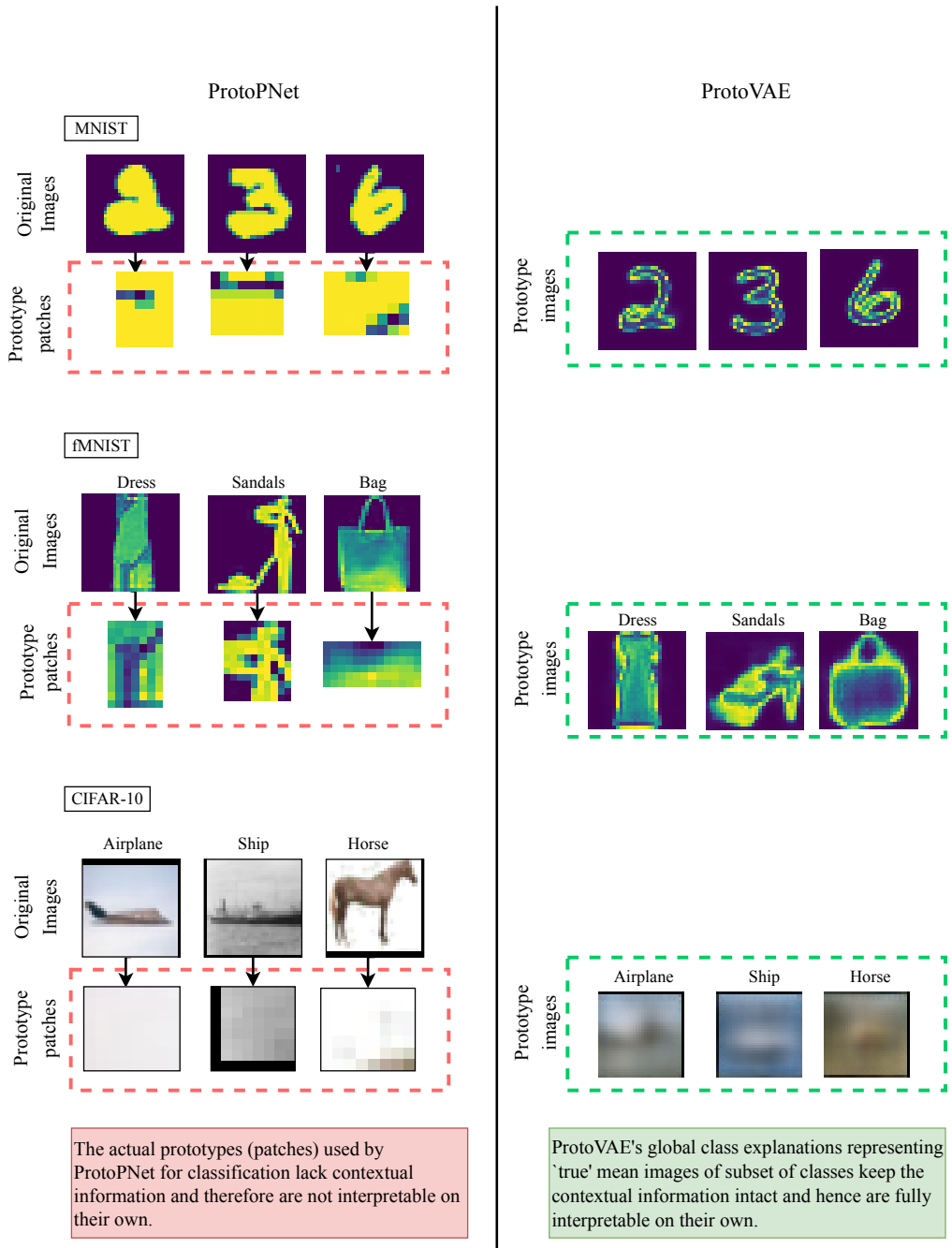


Figure 7: ProtoPNet vs ProtoVAE: Patch vs Image based global explanations.

(top left) has the positive relevance at the bottom of the aircraft for the horse prototype and positive relevance at the wheels and wheel-like objects as well as the sky for the automobile class prototype. The second test image (top right) has the head of the airplane resembling as an airplane and the shape near the wheels interestingly resembling a horse. The bottom left test image associates the shape of the ship with an airplane and a spherical shape resembling the wheel of an automobile. Finally, the bottom right test image has the front part of the vehicle activated by the automobile prototype and the top curved area resembling a ship.

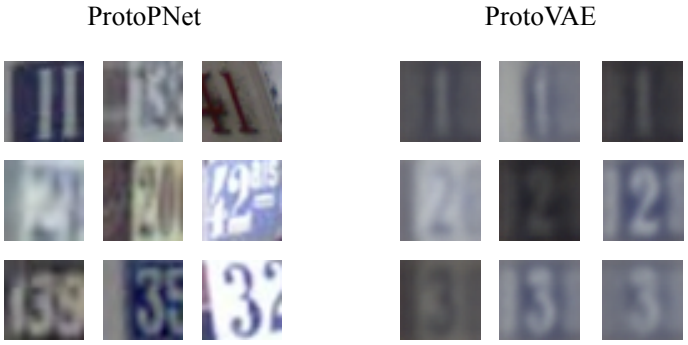


Figure 8: We show three learned prototypes for SVHN for ProtoPNet(left) and ProtoVAE(right). While the training data of SVHN consists of multiple digits in each image, ProtoPNet's prototypes visualizations also capture multiple digit images. However, due to the flexibility of our model, it has been observed that ProtoVAE always captures one digit, i.e., representing more faithful class prototypes.

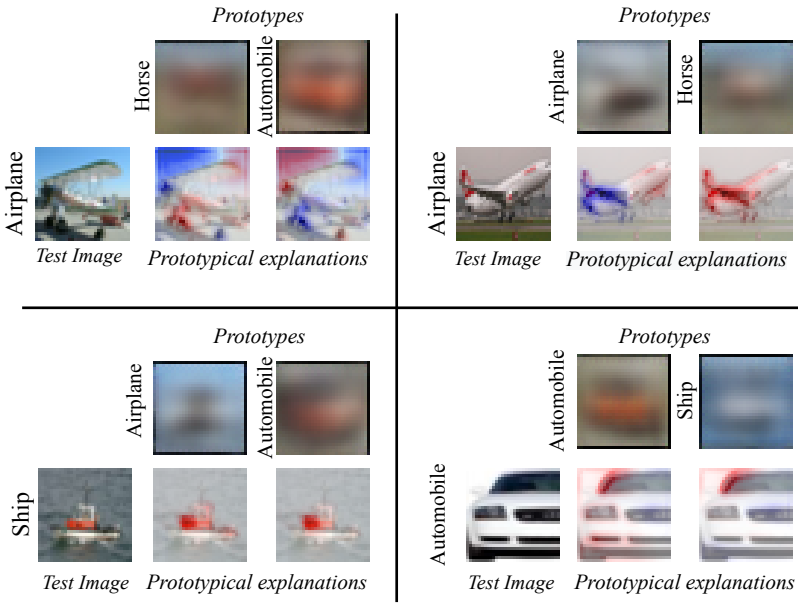


Figure 9: Prototypical explanations using LRP for CIFAR-10 dataset. For each test image, the prototypical explanations for 2 prototypes from different classes are provided.

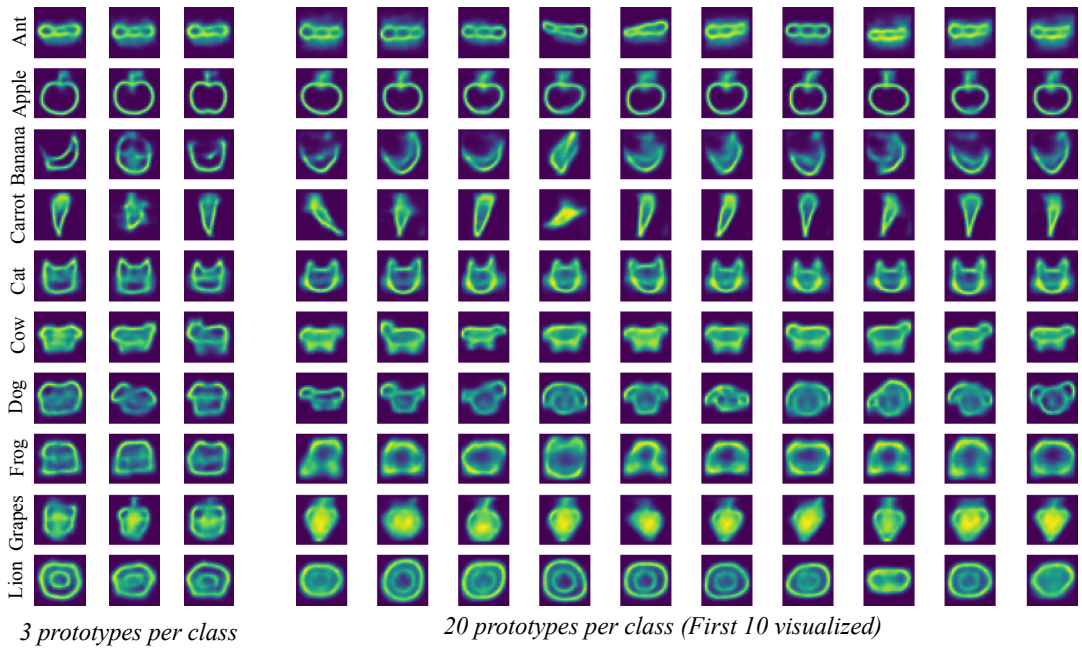


Figure 10: Effect on the sharpness of prototype reconstructions for QuickDraw when the number of prototypes per class are increased from 3 (top) to 20 (bottom). Sharpness can be observed for several classes (see Cat, Cow, Dog, Frog and Grapes, for example).

S6.8 Increasing number of prototypes per class

In this section, we demonstrate experimentally that the prototype reconstructions can get sharper as the number of prototypes per class increases. We show the results on QuickDraw (Figure 11) and CIFAR-10 (Figure 10 datasets). For QuickDraw, we show the difference between prototype reconstructions for ProtoVAE trained with 3 prototypes per class (Figure 11 left) and 20 prototypes per class (Figure 11 right). For CIFAR-10, we show the difference for models trained with 3, 5 and 20 prototypes per class (Figure 11). For both of the datasets the first 10 prototypes are visualized for the 20 prototypes per class trained models.

As observed for QuickDraw, we can see sharpness and added information in most of the classes. The largest effect of increased sharpness can be seen in classes Cat, Dog, Frog and Grapes. It can also be observed that the Dog prototypes are able to capture more variations of dogs, especially front view and side view as the number of prototypes increases. Similarly for CIFAR-10 we can see more variations of subsets of classes captured by prototypes when the number of prototypes increase thereby leading to less blurry prototype reconstructions. With only 3 prototypes per class, prototypes are degenerate and it is hard to guess that automobile prototypes depict cars. However, with 20 prototypes, we can see more orientations of cars captured by the model. The sharpness and variations captured by the prototypes can also be observed for other classes such as Dog, Horse etc.

S6.9 Effect of L2 norm on prototype reconstructions

To analyse the effect of the L2 norm, we demonstrate the prototypes learned with varying weight for the reconstruction loss term. The prototype reconstructions for QuickDraw for reconstruction loss weight 0.1 and 1 are shown in Figure 12 top and bottom, respectively and for 10 and 100 are shown in Figure 13 top and bottom, respectively. As can be clearly observed, the prototype reconstructions tend to be sharper with higher weight. This effect can be observed in all classes. For example, for the class Cat we start to see clear boundaries as well as whiskers when the weight is 100. We can further observe sharper grapes, dogs as well as frogs when we move from weight 0.1 to 100. Similarly, for CIFAR-10, we can observe more details present in the prototype reconstructions as the weight for the reconstruction loss is increased from 0.1 to 10 (see Figure 14. We can observe different colors and

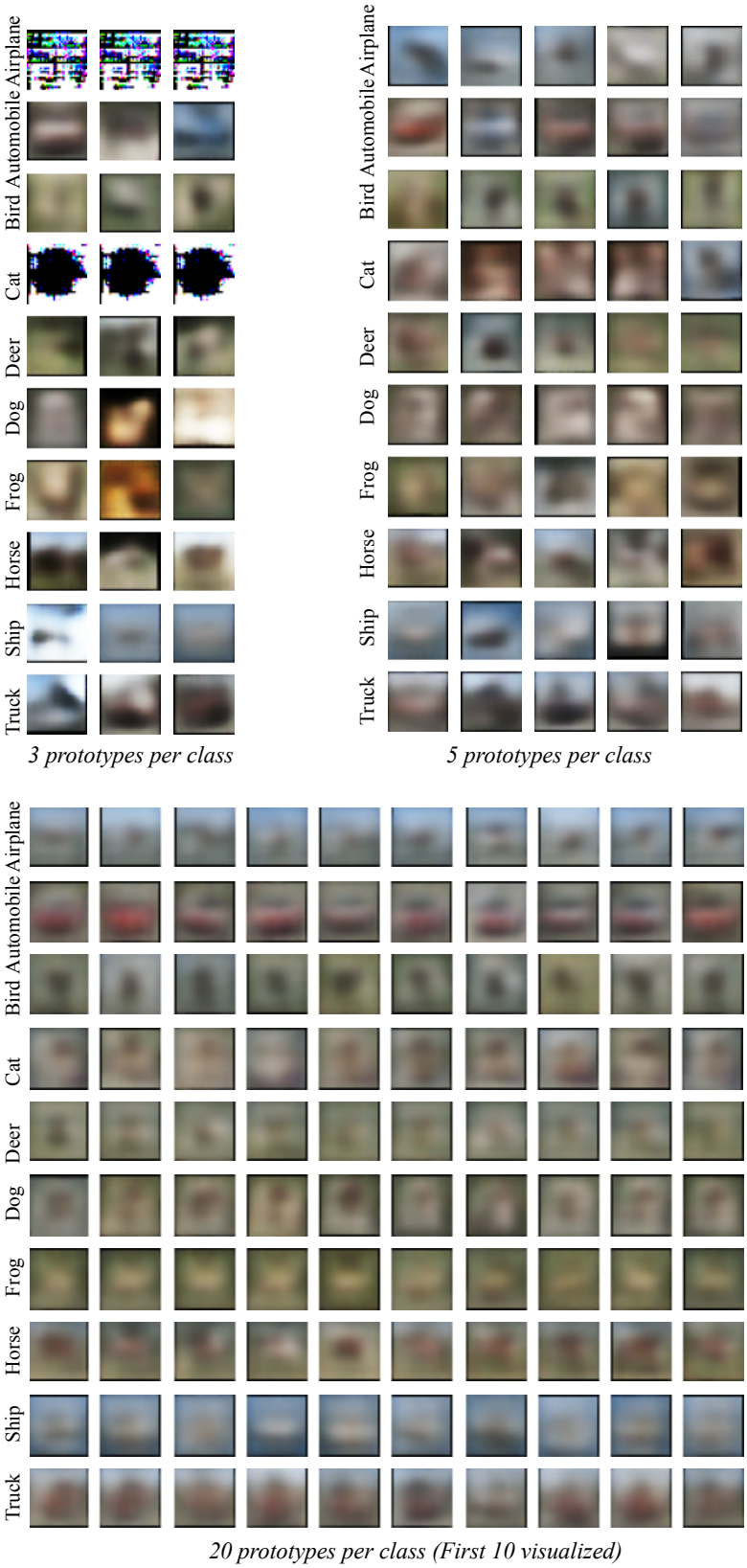
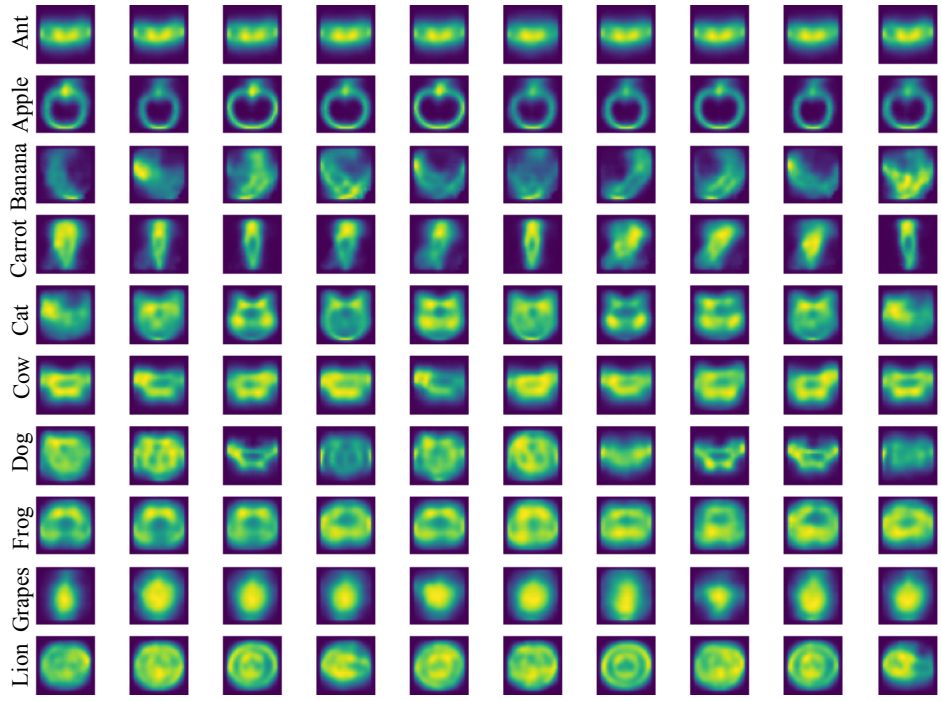
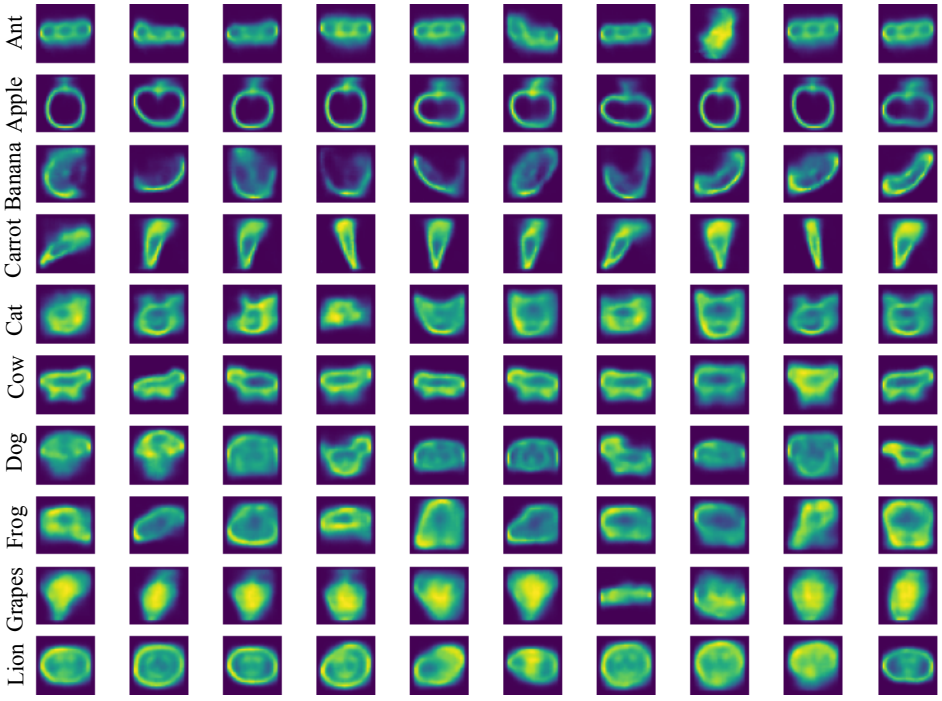


Figure 11: Effect on the sharpness of prototype reconstructions for CIFAR-10 for the models trained with 3, 5 and 20 number of prototypes per class. Less blurriness and more variations captured can be observed for several classes (see Automobile, Horse, Ship, for example).

orientations of automobiles captured as well as comparatively more detailed airplanes, horses and trucks for higher weight of the L2 norm.

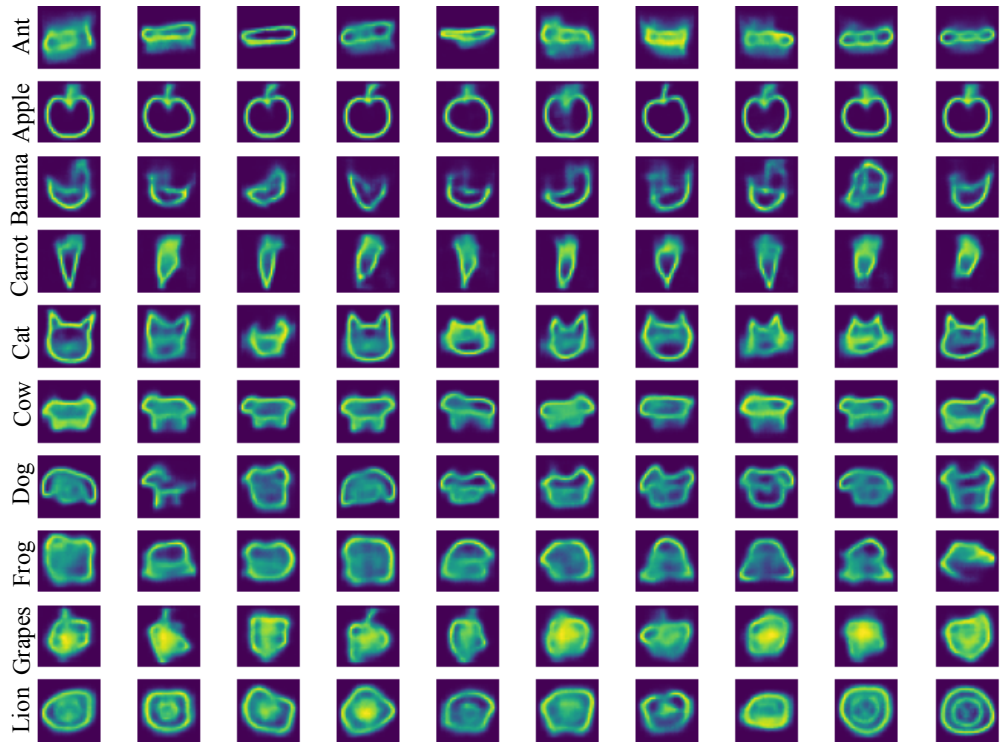


Reconstruction loss weight=0.1

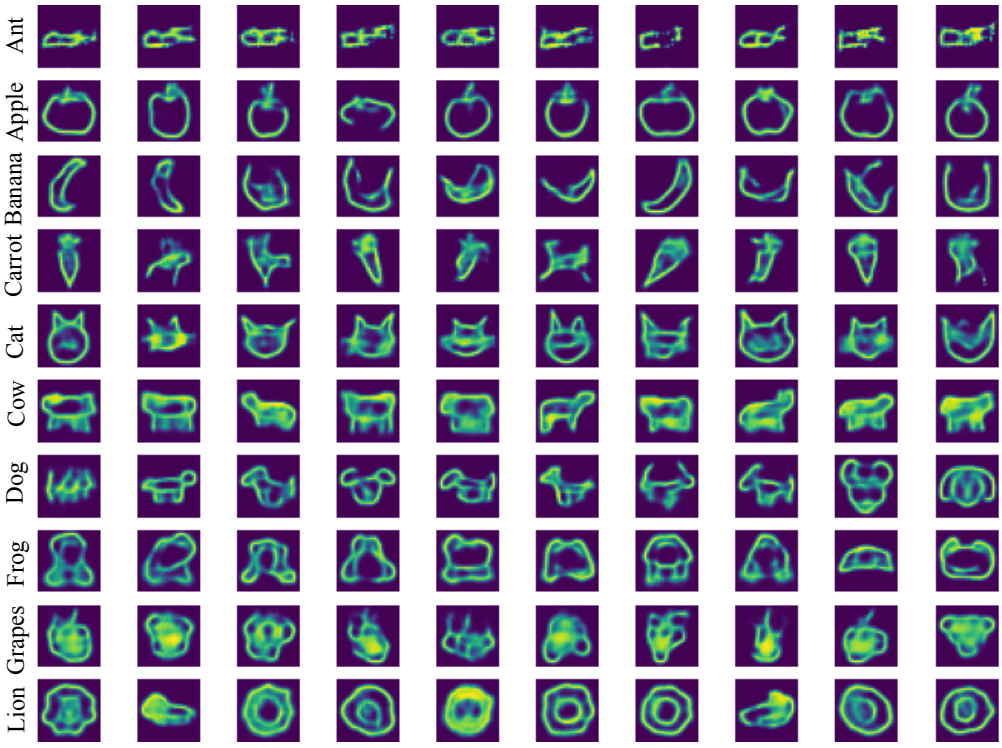


Reconstruction loss weight=1

Figure 12: Prototypes for models trained on QuickDraw dataset with reconstruction loss term weighted 0.1 (top) and 1 (bottom).



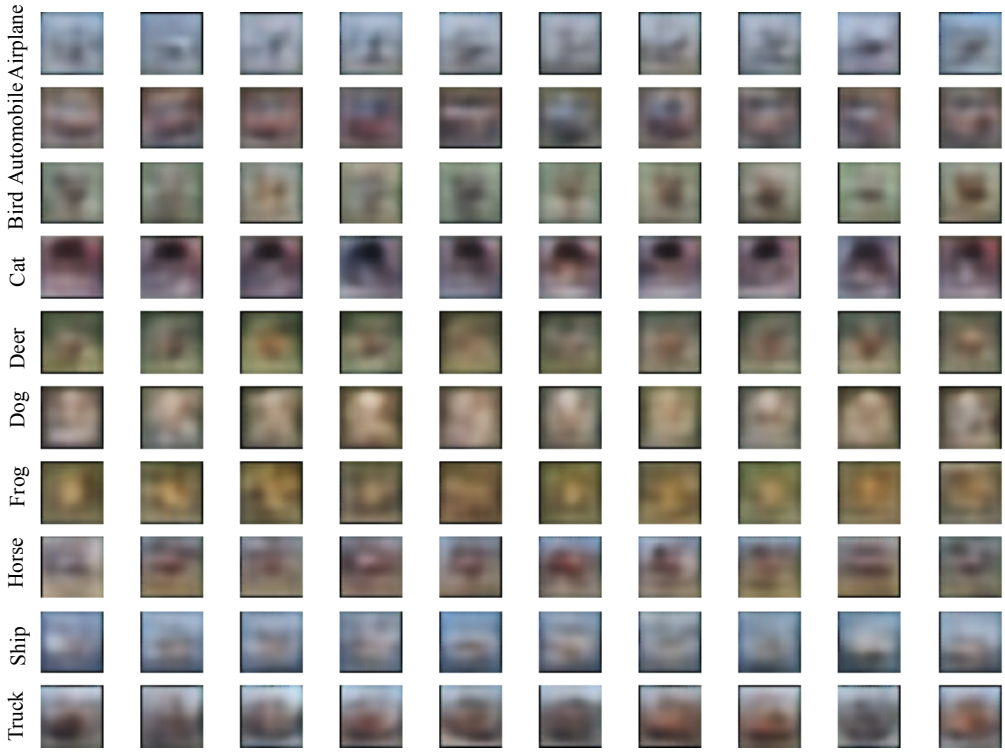
Reconstruction loss weight=10



Reconstruction loss weight=100

Figure 13: Prototypes for models trained on QuickDraw dataset with reconstruction loss term weighted 10 (top) and 100 (bottom). Sharper prototypes are generated with higher weight of the reconstruction loss.

Reconstruction loss weight=0.1



Reconstruction loss weight=10

Figure 14: Prototypes for models trained on CIFAR-10 dataset with reconstruction loss term weighted 0.1 (top) and 10 (bottom). Less blurry and comparatively more detailed prototypes can be observed with higher weight.

S6.10 ProtoVAE with high resolution images

To demonstrate the applicability of ProtoVAE on more complex real world datasets and high resolution images, we report here an experiment on the CelebA dataset [7]. The objective is to demonstrate the scalability, quality and diversity of the learned prototypes, and the classification task is to distinguish males and females. The images have originally a resolution of 178×218, but are scaled to 224×224 for processing. The backbone consists of a ResNet-18 encoder followed by a decoder that resembles the architecture in Fig. 1 but is extended to 224×224 output resolution. The different sub-losses $\mathcal{L}_{\text{CE}}$, $\mathcal{L}_{\text{VAE}}$ and $\mathcal{L}_{\text{orth}}$ are weighted by 1, 1.000 and 10, respectively. The model is trained to learn 20 prototypes per class, which are depicted in Fig. 15. The test accuracy is 98.2%.

While the images have a blurry contour, several attributes clearly vary. For the women, we can distinguish different hair style, length and color, more or less open smiles, rosy cheeks and the age (second line, third and fourth from the left seem older). Interestingly, no female prototypes wear sunglasses, while the first male prototypes of the second row does. Regarding males, the face orientation seems more constant. The color of the skin and of the hair also vary. The second prototype in the first row seems bald. Despite the high accuracy, an anomaly seems to be present in form of the fourth male prototypes of the first row. The analysis of the embedding indicates that a male prototype is present within the female neighborhood. We assume that this could help to better model men with long hair, as they are not represented by other male prototypes.

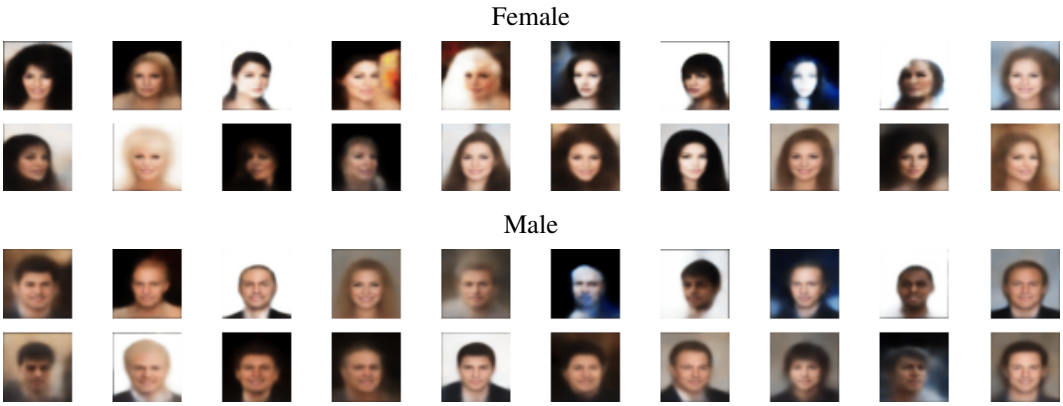


Figure 15: Prototypes learned for CelebA for ResNet-18.

S6.11 Reconstruction of test images with ProtoVAE

Since the prototype reconstructions can only be as good as the image reconstructions, we provide the test image reconstructions generated by ProtoVAE for MNIST, fMNIST, CIFAR-10, QuickDraw and SVHN in Figure 16. As can be seen, our model generates crisper reconstructions for MNIST, fMNIST, QuickDraw and SVHN as compared to CIFAR-10, which echoes the sharpness of the prototype reconstructions of Fig. 2 in the main document.

S7 Negative Societal Impacts

The proposed method will have positive societal impacts by enabling transparency while obtaining similar performance to the black-box counterparts. Nonetheless, it is still susceptible to adversarial attacks [8] similar to other existing deep learning models. A thorough inspection of the class-prototypes is thus required before it can be leveraged in safety critical scenarios to avoid spurious learning [9] or backdoor triggers [10].

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing.

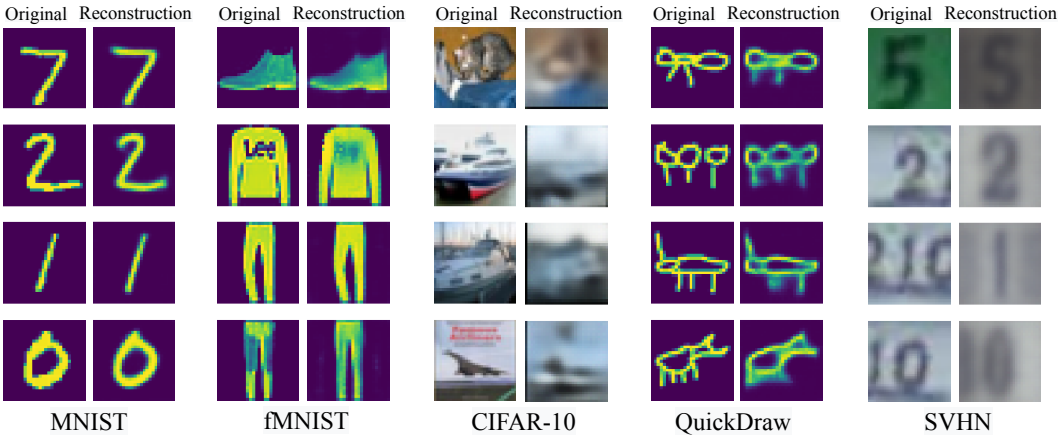


Figure 16: Visualization of test image reconstructions for MNIST, fMNIST, CIFAR-10, QuickDraw and SVHN.

[2] Jayneel Parekh, Pavlo Mozharovskyi, and Florence d'Alché-Buc. A framework to learn with interpretation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24273–24285. Curran Associates, Inc., 2021.

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[4] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[7] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[8] Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *ArXiv*, abs/2007.10760, 2020.

[9] Christopher J. Anders, Talmaj Marinc, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Analyzing imagenet with spectral relevance analysis: Towards imagenet un-hans'ed. *ArXiv*, abs/1912.11425, 2019.

[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *SafeAI@AAAI*, 2019.