Robust Hierarchical Symbolic Explanations in Hyperbolic Space for Image Classification: Supplementary Material

1. Hyperbolic mappings

We show here the general equations $exp_x^K(y)$ and $log_x^K(v)$ with any negative curvature, $-1/K$ for $\mathbb{H}^{d,K}$ in Eq. (1) and Eq. (2) respectively as well as for $\mathbb{B}^{d,K}$ in Eq. (3) and Eq. (4) respectively [1, 2].

1.1. Hyperboloid

$$exp_x^{\mathbb{H},K}(y) = cosh\left(\frac{\|y\|_S}{\sqrt{K}}\right) x + \sqrt{K} sinh\left(\frac{\|y\|_S}{\sqrt{K}}\right) \frac{y}{\|y\|_S} \tag{1}$$

$$log_x^{\mathbb{H},K}(v) = d^{\mathbb{H},K}(x,v)\left(\frac{v + \frac{1}{K}\langle x, v\rangle_S \, x}{||v + \frac{1}{K}\langle x, v\rangle_S \, x||_S}\right) \tag{2}$$

1.2. Poincare

A Riemmanian metric tensor is conformal to another Riemannian metric if it defines the same angles. In the case of the Poincare unit ball, there is a smooth conformal mapping, $\lambda : \mathbb{B} \to \mathbb{R}$ between the Euclidean metric tensor $g_x^{\mathbb{R}}$ and Poincare metric tensor $g_x^{\mathbb{B},1}$. The Poincare conformal factor $\lambda_x = \frac{2}{1-\|x\|^2}$ is defined such that $g_x^{\mathbb{B},1} = \lambda_x^2 g_x^{\mathbb{R}}$.

$$exp_x^{\mathbb{B},K}(y) = x \oplus^K \left(tanh(\frac{\sqrt{K}\lambda_x^K \|y\|}{2})\frac{y}{\sqrt{K}\|y\|}\right) \tag{3}$$

$$log_x^{\mathbb{B},K}(v) = \frac{2}{\sqrt{K}\lambda_x^K} arctanh(\sqrt{K}\|-x \oplus^K v\|)\frac{-x \oplus^K v}{\|-x \oplus^K v\|}) \tag{4}$$

2. Parallel transport

In this section, we reveal the equations of parallel transport for vector b on the tangential space of the origin to the tangential space of v. Formally, we show $P_{o\to v}^K(b)$ in $\mathbb{B}^{K,d}$ and $\mathbb{H}^{K,d}$ below.

2.1. Hyperboloid

The general form of this equation, $P_{x\to v}^K(b)$ shown below in Eq. (5) is derived in [1].

$$P_{o\to v}^K(b) = b - \frac{\langle log_o(v), b\rangle_S}{d^{\mathbb{H},1}(o,v)^2}(log_o(v) + log_v(o)) \tag{5}$$

2.2. Poincare

Eq. (6) is derived in [2].

$$P_{o\to v}^K(b) = \frac{\lambda_o^K}{\lambda_v^K} b \tag{6}$$

3. Projections

We need to apply projections to constrain points to the manifolds and its tangential space after optimising. In our work, all operations are performed in the tangential plane and therefore one needs to only apply projections after the exponential mapping from the tangential space to the manifold which we define below.

Eq. (7), below shows the projection after exponential mapping y on $\mathcal{T}_x\mathbb{H}^{d,K}$ to the hyperboloid manifold $\mathbb{H}^{d,K}$. In our work we use $K = 1$ for applying projections.

$$Proj(exp_x^{\mathbb{H},K}(y)) = (\sqrt{1 + ||v_{1:d}||_2^2}, v_{1:d}) \tag{7}$$

The projection when mapping a point y on $\mathcal{T}_x\mathbb{B}^{d,K}$ to Poincare space $\mathbb{B}^{d,K}$ is achieved by normalising $exp_x^{\mathbb{B},K}(y)$ if $||exp_x^{\mathbb{B},K}(y)||_2^2 > 1/\sqrt{K}$ or > 1 in the case of the Poincare unit ball.

4. Training details and architectures

4.1. Discrete surrogate model

We consider an image dimension of 32×32 for the MNIST datasets. The image dimensions for the AFHQ and STL10 dataset is 128×128. The image dimensions for the MIMIC dataset is 224×224. Our modulation layer reduces the number features (latent dimension) of z from 64 to 8 channels and 1024 to 32 channels for the MNIST and AFHQ experiments respectively. We perform all hyperbolic linear operations in $\mathcal{T}_x\mathbb{H}$ and do not directly update the embeddings $\zeta^i \in \mathbb{H}^{1,d}$ or $\in \mathbb{B}^{1,d}$ which means

we can perform Euclidean optimisation. Therefore, for the experiments in the main paper, we use Adam optimisation with a learning rate of 0.0002 and batch size of 50 in all experiments to train the discrete surrogate model for 40 epochs on a single NVIDIA RTX 2080 GPU. We experimented with Riemmanian stochastic gradient descent [11] but found Adam optimisation to be more stable. We also found that $\zeta^n \in \mathbb{H}^{M_i \times d,1}$ provided greater stability during training and showed generally improved performance (see codebook ablation Tab. 1 and Tab. 2 below in Sec. 5) Note, we do not perform linear operations in $\mathcal{T}_x\mathbb{B}$ due to stability issues. Also, note we do not directly update the embeddings of ζ^i in $\mathbb{H}^{1,d}$ or $\mathbb{B}^{1,d}$ as the embeddings are updated only as a function of the learned hyperboloid linear operations in $\mathcal{T}_x\mathbb{H}$ followed by a mapping to $\mathcal{T}_x\mathbb{B}$.

Furthermore, it is important state, that low dimensional embeddings for ζ^0, for example where $d = 2$ will have an adverse effect on the output of the decoder, producing very blurry images. In this case, one can use the Euclidean codebook with higher dimensionality d for and theof The only embeddings we update directly through vector quantisation are in the the Euclidean codebook $\mathbb{E} \in \mathbb{R}^{d' \times M_0}$. [1] applies hyperboloid non-linear activation to aid with the smoothly varying curvature. We do not include non-linearity in our hyperboloid linear layer which uses fixed negative curvature as it did not add any benefit to training convergence or knowledge distillation performance.

4.2. Binary Weights

We described the optimisation of the Binary weights in the main paper using the *boop* [4] algorithm. In this algorithm, the strength of gradient signal at time t is determined by looking at the continuous exponential moving average m_t of accumulated gradients up to the gradient, g_t at t. The binary weights are then updated subject to m_t exceeding a threshold τ and the sign of $w_{t,l}^{j,k}$ matching m_t. We initialize weights randomly $\in \{0, 1\}$ and modify the update rule proposed in [4]. This is shown in Equation 8 below, where the first line corresponds to calculating continuous exponential moving average with γ being the adaptivity rate [4] and the second line defines the update rule [4]. By following this update rule, we ensure that weights are either 0 or 1 while at the same time, following all the properties described in [4].

$$
\begin{aligned}
m_t &= (1-\gamma)m_{t-1} + \gamma g_t \\
w_{t,l}^{j,k} &= \begin{cases} |w_{t-1,l}^{j,k} - 1|, & \text{if } |m_t^i| > \tau \text{ and} \\ & \text{sign}(m_t^i) = \text{sign}(w_{t-1,l}^{j,k}) \\ w_{t-1,l}^{j,k}, & \text{otherwise} \end{cases}
\end{aligned} \tag{8}
$$

In our experiments, we use, adaptivity rate γ of 0.0004 and threshold τ of $1e-8$. The adaptivity rate is analogous to the learning rate and can be seen as the consistency of a gradient signal required to induce a flip in weights from 0 to 1 or vice versa [4]. A high γ can therefore induce a flip quicker given a new gradient signal but this can also mean noisy training [4]. τ is reflective of the strength of the gradient signal required to induce a flip [4].

4.3. Decoder

The Decoder for MNIST experiments initially consists of a 1×1 convolutional block to increase the number of channels back to 64 channels before 3 up-sampling blocks. Each up-sampling block consists of bi-linear interpolation before two pre-activation 3×3 convolutional blocks with residual connections [3]. We use batch normalisation and ReLU activation. In our AFHQ,STL10 and MIMIC experiments, the decoder also initially starts with a 1×1 convolutional block to increase the number of channels back to 1024 before 4 up-sampling blocks identical to the one used in our MNIST experiments. We train the decoder in both experiments for 40 epochs using Adam optimisation with a learning rate of 0.0002.

4.4. Pre-trained classifiers

The MNIST pre-trained classifier consists of 7 convolutional blocks made up of a 3×3 convolutional layer followed by batch normalisation and ReLU non-linearity. This is followed by global average pooling and a single layer linear classifier. Max pooling is applied after the first, third and fifth layers. The number of channels corresponding to the final convolutional block is 64. We train this classifier for 50 epochs with a batch size of 50 on three NVIDIA RTX 2080 GPUs. We use Adam optimisation with an initial learning rate of 0.001 and weight decay of 0.001. We achieve 99% accuracy with this pre-trained classifier

We use the DenseNet-121 as our pre-trained classifier for the AFHQ,STL10 and MIMIC dataset [5]. We train using Adam optimisation with an initial learning rate of 0.001 and weight decay of 0.05. We reduce the dimensionality of the MNIST, AFHQ, STL10 and MIMIC images to 4×4 in the final layer before being inputted into the classifier.

4.5. Optional commitment loss and Uncertainty

4.5.1 Optional commitment loss

Given $z_{pq}^i \in \mathbb{R}^{K \times d'}$ and $k \in K$, we define an optional commitment loss to move each $z_{pqk}^i \in z_{pq}^i$ closer to its sampled codebook vector denoted ζ_k^i shown below in Eq. (9).

$$
\mathcal{L}_{cb} = \sum_{i=0}^{i=n} \sum_{t=0}^{t=K} d^{\mathbb{B},1}(z_{pqk}^i),\ sg(\zeta_k^i)). \tag{9}
$$

Therefore the total loss in this case is: $\mathcal{L}_{Total} = \mathcal{L}_{dist} + \mathcal{L}_{quant} + \mathcal{L}_{Poincare} + \epsilon\mathcal{L}_{cb},\ \epsilon \in \{0, 1\}$.

Table 1. Knowledge distillation accuracy of different codebook sizes for Poincare, Hyperboloid and Euclidean embeddings on the MNIST dataset. The best knowledge distillation accuracy for each embedding dimension is highlighted in bold.

Embedding dim. (→)	**Poincare**			**Hyperboloid**			**Euclidean**		
Codebook Size ($\zeta^0, \zeta^1, \zeta^2$) (↓)	**2**	**4**	**16**	**2**	**4**	**16**	**2**	**4**	**16**
512, 64, 4	0.92	0.91	0.98	**0.95**	**0.93**	**0.99**	0.80	0.90	0.95
256, 64, 4	**0.94**	0.94	**0.99**	0.91	**0.98**	**0.99**	0.86	0.92	0.96
256, 32, 4	0.89	0.96	**0.99**	**0.93**	**0.98**	0.97	0.79	0.91	0.98
128, 32, 4	0.90	**0.96**	**0.99**	**0.91**	0.95	**0.99**	0.81	0.92	0.95
64, 16, 4	0.85	**0.88**	**0.94**	**0.90**	**0.88**	0.93	0.80	0.83	0.90

Table 2. Knowledge distillation accuracy of different codebook sizes for Poincare, Hyperboloid and Euclidean embeddings on the AFHQ dataset. The best knowledge distillation accuracy for each embedding dimension is highlighted in bold.

Embedding dim. (→)	**Poincare**			**Hyperboloid**			**Euclidean**		
Codebook Size ($\zeta^0, \zeta^1, \zeta^2$) (↓)	**2**	**4**	**16**	**2**	**4**	**16**	**2**	**4**	**16**
512, 64, 2	**0.94**	0.92	0.96	0.90	**0.95**	**0.98**	0.83	0.91	0.93
256, 64, 2	0.90	0.95	0.98	**0.92**	**0.98**	**0.99**	0.80	0.90	0.97
256, 32, 2	0.86	0.85	0.93	**0.88**	**0.91**	**0.96**	0.77	0.81	0.88
128, 32, 2	**0.88**	0.88	0.91	0.84	**0.92**	**0.96**	0.80	0.86	0.90
64, 16, 2	0.78	0.85	0.83	**0.81**	**0.90**	**0.92**	0.75	0.80	0.84

We found this to not affect knowledge distillation performance except to provide a more certain notion (reduced uncertainty) of abstraction for z_q^i

4.5.2 Uncertainty

Given we sample by distance, this allows to capture uncertainty over a single FOL chain rule defined as the likelihood of the sampled chain rule over all possible chain rules that can be sampled to classify a single class. We must firstly define the number of possible symbols which can be sampled from each codebook to define the total number of chain rules. In the first codebook ζ^0, all possible symbols can be sampled and therefore for the first symbol sampled in the chain rule we can define uncertainty over the first layer of abstraction as the exponentiated Poincare distance from z_{pqk}^0 to the sampled symbol ζ_j^0 normalised over the total number symbols in ζ^0 divided by a length-scale (σ) hyperparameter. This is equivalent to using the radial basis function for measuring uncertainty which we adapt from [10]. For the following codebooks, we define the possible symbols which can be sampled from ζ^i to be those with edges from codebook ζ^{i-1}. We repeat the same process as we did to capture uncertainty over the first layer of abstraction for the following symbols in the FOL chain rule and then multiply the uncertainties measured for each sampled symbol in the FOL chain rule; this will therefore determine uncertainty for the sampled chain rule. We formally define the uncertainty of a sampled chain rule (CR) in Equation 10 below. We define the total number of available symbols which can be sampled from each codebook as S^i

$$Uncertainty_{CR} = \prod_{i=0}^{n} exp\left[\frac{\frac{1}{S^i} d^{\mathbb{B},1}(z_{pqk}^i, \zeta_j^i)}{\sigma}\right] \tag{10}$$

5. Codebook ablations

We determine through our ablations, the minimum number of concepts at each level of the tree for each experiment shown in Tab. 3. We show detailed codebook ablations to find the minimum number of codebook vectors required to achieve a knowledge distillation accuracy of at least 90% for MNIST in Tab. 1 and AFHQ in Tab. 2. We limit the number of codebook vectors in ζ^n to at least $\lfloor log_2 N + 1 \rfloor$.

	MNIST	AFHQ	STL10	MIMIC
ζ^1	128	256	256	256
ζ^2	32	64	64	64
ζ^3	4	3	4	2

Table 3. Codebook ablations to develop the most sparse knowledge tree

We note in more complex datasets, a larger number of codebook vectors maybe required in ζ^n, whereby a class may be need to be encoded with more than one possible combination of codebook vectors in order to fit the data.

We also compare knowledge distillation accuracy of codebooks with Poincare embeddings against hyperboloid embeddings. In these experiments we only use a three level hierarchy. The user can however decide if they would like more levels of abstraction by increasing the number of codebooks. We note also, the use of codebook ablations is a limitation for the practicality of our explainability method.

6. Robustness experiments

We show that there is significantly less variance in the explanations generated by our method with the addition of Poisson noise in Tab. 4 and S&P noise in Tab. 5 highlighting the robustness of our explanations. The close to 0 variances also reflects the image level trees not changing under different perturbations and shows the robustness of the rules derived.

	MNIST	STL10	MIMIC	AFHQ
LIME [7]	0.206	0.747	0.577	1.132
SHAP [6]	0.147	0.611	0.483	1.943
deepLIFT [9]	0.458	1.173	0.563	1.477
gradCAM [8]	0.966	1.100	1.074	1.094
Ours	$\mathbf{1e^{-6}}$	$\mathbf{1e^{-5}}$	$\mathbf{1e^{-5}}$	$\mathbf{1e^{-5}}$

Table 4. Average variance in the heatmaps generated by various post-hoc explainability methods under Poisson noise addition.

	MNIST	STL10	MIMIC	AFHQ
LIME [7]	0.785	1.342	1.178	1.157
SHAP [6]	1.911	13.124	6.746	14.494
deepLIFT [9]	0.935	6.759	5.903	3.185
gradCAM [8]	52.472	66.441	19.664	55.488
Ours	$\mathbf{8e^{-7}}$	$\mathbf{2e^{-6}}$	$\mathbf{1e^{-5}}$	$\mathbf{1e^{-5}}$

Table 5. Average variance in the heatmaps generated by various post-hoc explainability methods under Salt and Pepper (S&P) noise addition.

We show a visual example of the robustness of our method under 30% Gaussian noise for a MNIST dataset example in Fig. 1. We also show a visual example of the robustness of our method under 30% salt and pepper noise for a MIMIC dataset example in Fig. 2 (same example as used in the main paper). We next show a visual example of the robustness of our method under 20% Poisson noise for a STL10 dataset example in Fig. 3

7. Additional results

In this section, we provide additional examples illustrating the explanations of the proposed framework. Fig. 5, Fig. 6 and Fig. 7 are explanations for a model trained on the MNIST dataset classifying class '3', '9' and '6' respectively. We show the class level tree for the dog class in AFHQ in Fig. 8. Fig. 9 and Fig. 10 describes our explanations for a model trained on the AFHQ dataset classifying the class 'dog' and 'cat' respectively. Fig. 11 shows explanations for a model trained on the STL10 dataset for the class 'bird'. Fig. 12 shows explanations for a model trained on the MIMIC dataset for the class 'abnormal'.

Fig. 4 demonstrates the distribution of codebook symbols on a Poincare disk for the AFHQ dataset.

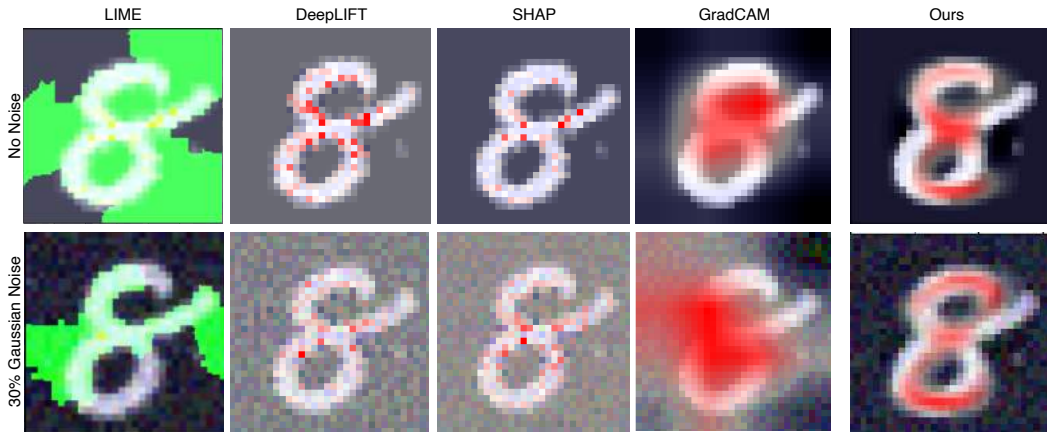


Figure 1. Robustness experiments with 30% Gaussian noise on a sample from the MNIST dataset. Symbol for ζ_3^2 is shown. The image level tree did not change under 30% Gaussian noise for this example.

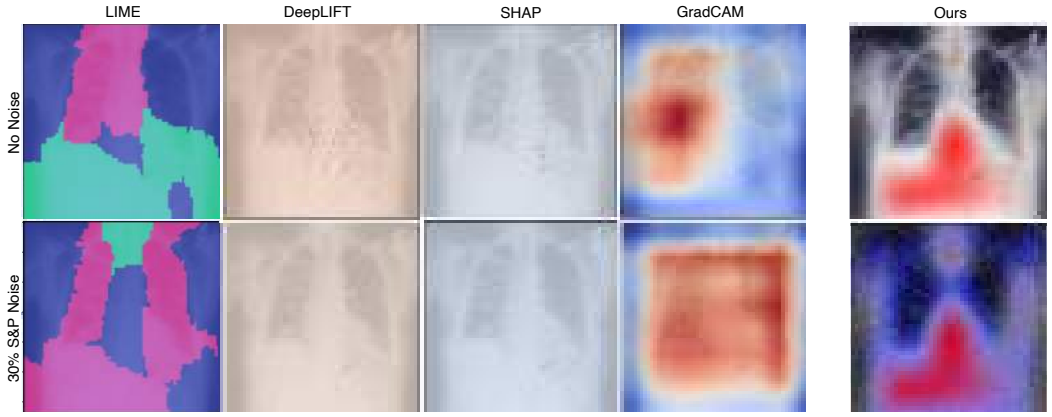


Figure 2. Robustness experiments with 30% salt and pepper noise on a sample from the MIMIC dataset. Symbol for ζ_0^2 is shown. The image level tree did not change under 30% S&P noise for this example.

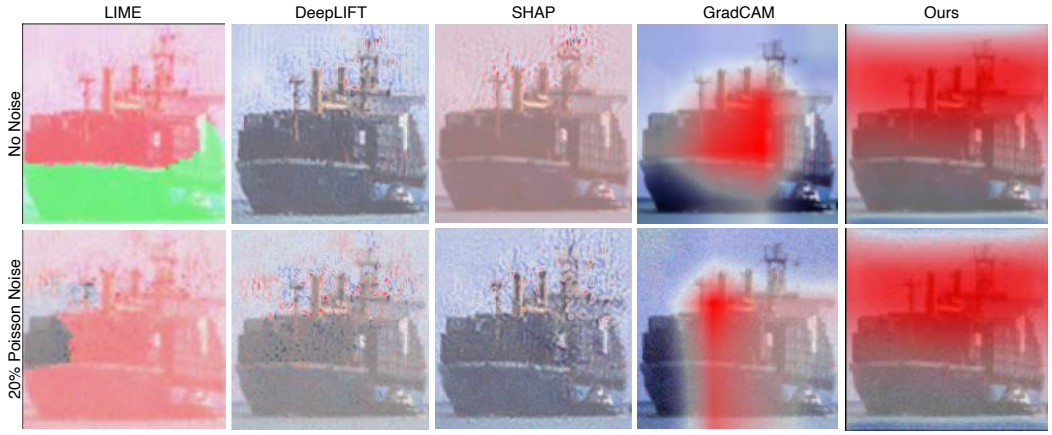


Figure 3. Robustness experiments with 20% Poisson noise on a sample from the STL10 dataset. Symbol for ζ_0^2 is shown. The image level tree did not change under 20% Poisson noise for this example.

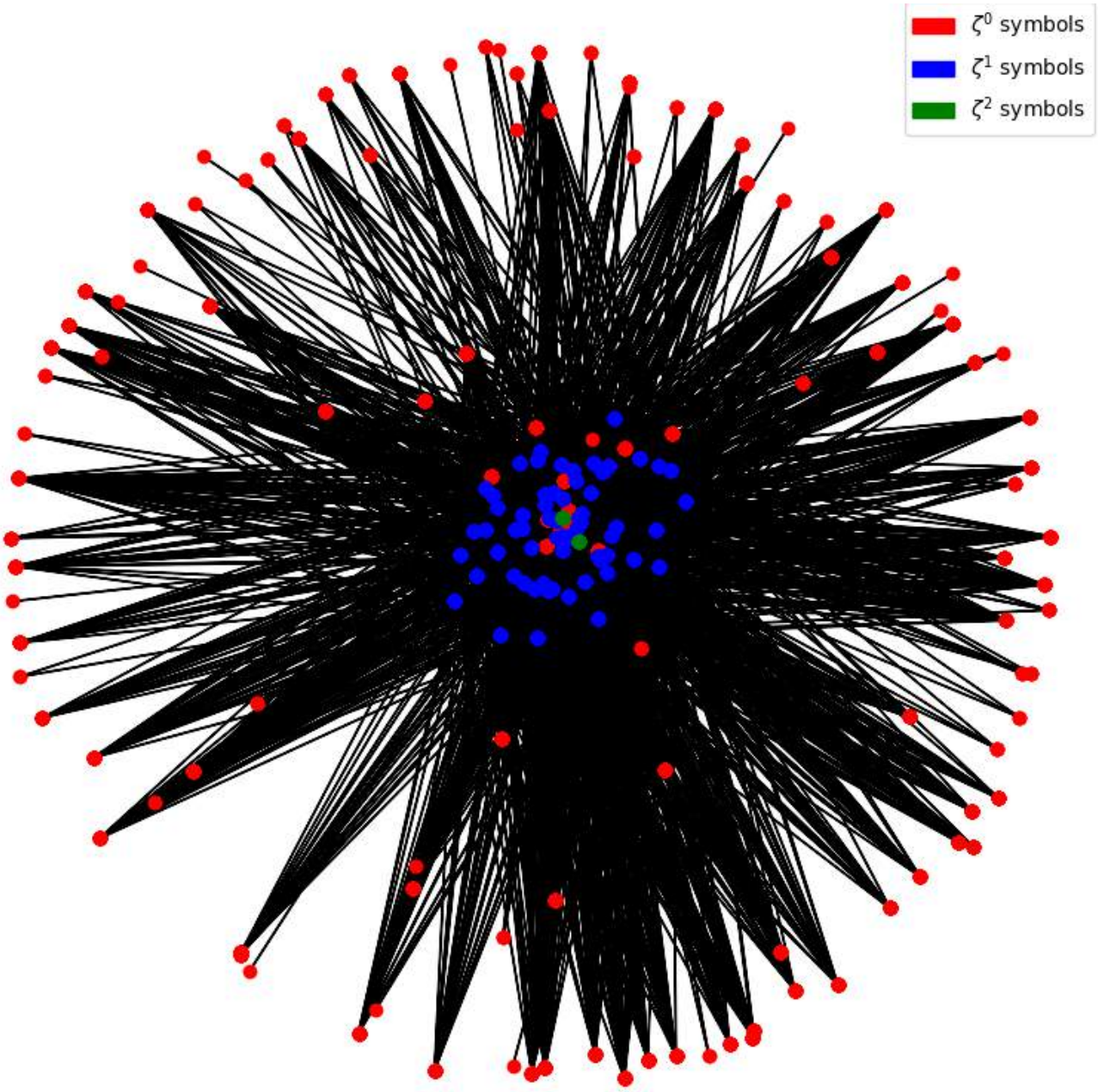
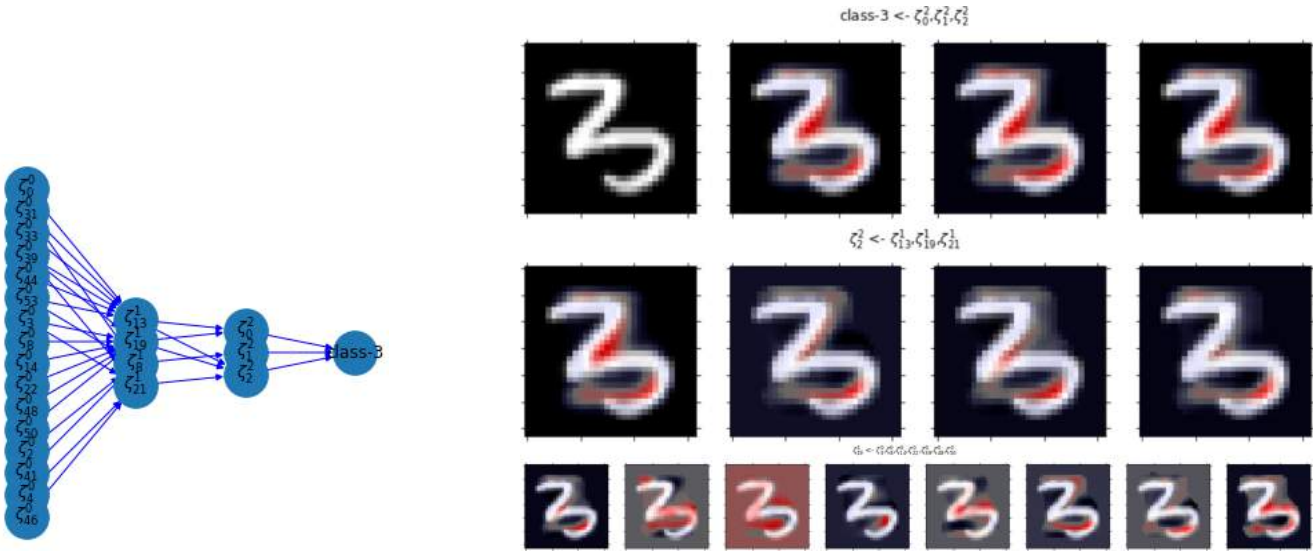
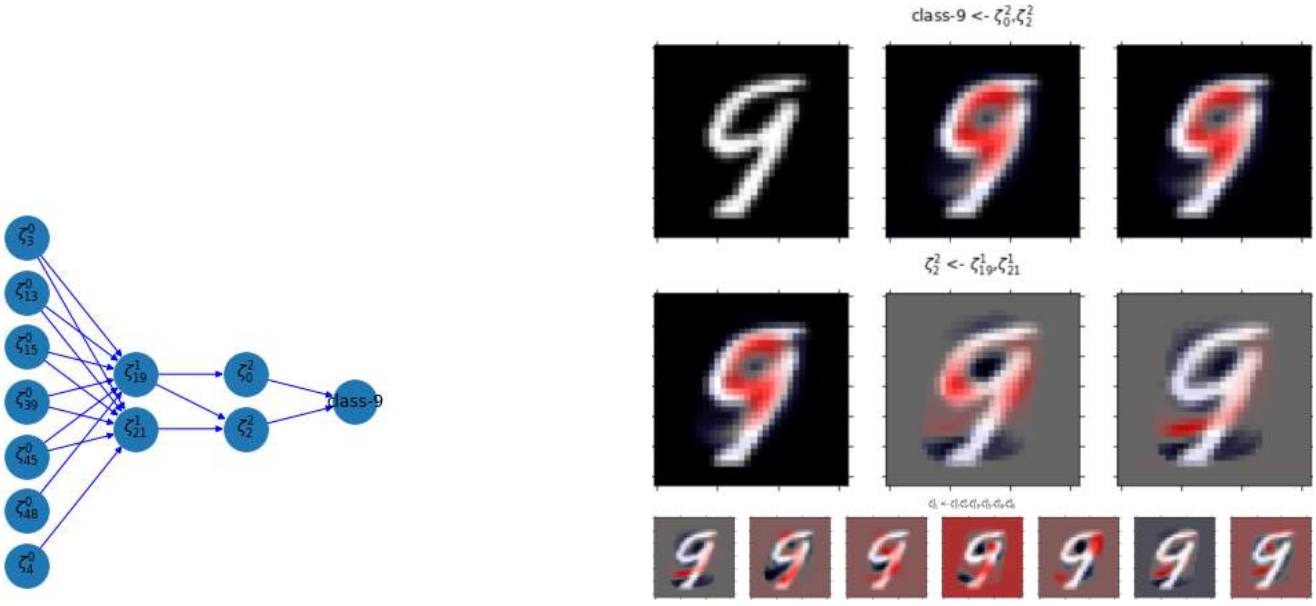


Figure 4. Poincare embedding of discrete symbols obtained for the AFHQ classifier. Here, red, blue, and green nodes indicate symbols from $\zeta^0, \zeta^1, \zeta^2$ layers abstraction respectively.



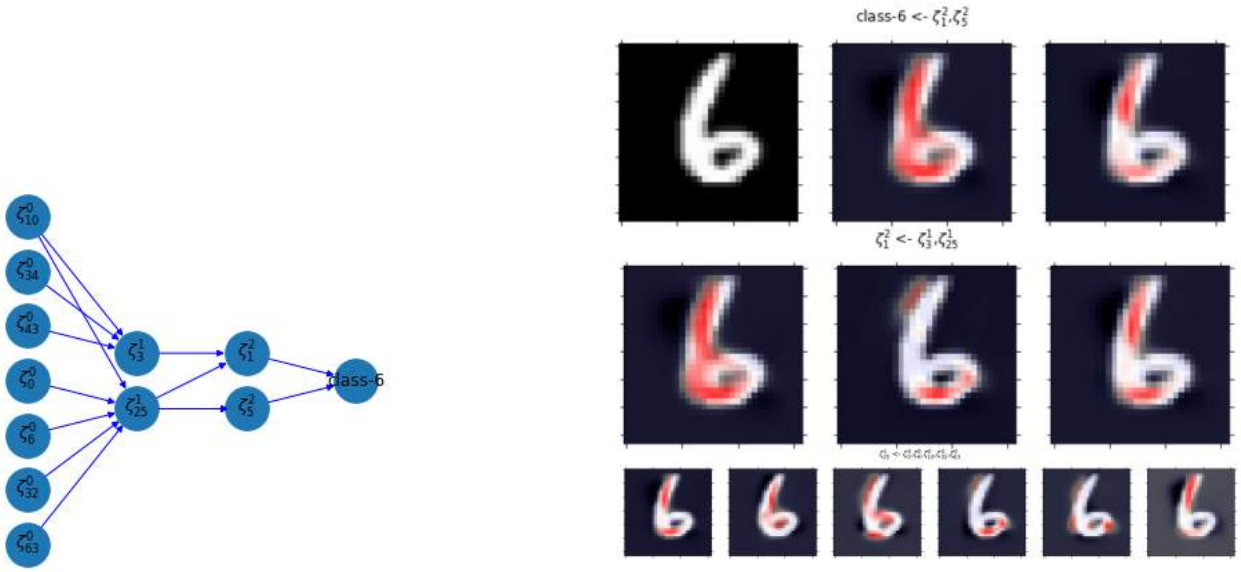
(a) Image-level tree (b) Visual rules

Figure 5. This figure describes the hierarchical visual explanations obtained using our proposed framework on the MNIST classifier for class '3', with atoms corresponding to $pf(Class3, \zeta_0^2), pf(Class3, \zeta_1^2), pf(Class3, \zeta_2^2)$ described in the first row. The second row visualises the obtained atoms corresponding to $pf(\zeta_2^2, \zeta_{13}^1), pf(\zeta_2^2, \zeta_{19}^1), pf(\zeta_2^2, \zeta_{21}^1)$. The last row visualises the atoms corresponding to symbols in ζ^0 which are part of ζ_{19}^1.



(a) Image-level tree (b) Visual rules

Figure 6. This figure describes the hierarchical visual explanations obtained using proposed framework on the MNIST classifier for class '9', with atoms corresponding to $pf(Class9, \zeta_0^2), pf(Class9, \zeta_2^2)$ described in the first row. The second row visualises the obtained atoms corresponding to $pf(\zeta_2^2, \zeta_{19}^1), pf(\zeta_2^2, \zeta_{21}^1)$. The last row visualises the atoms corresponding to symbols in ζ^0 which are part of ζ_{21}^1.



(a) Image-level tree

(b) Visual rules

Figure 7. This figure describes the hierarchical visual explanations obtained using our proposed framework for the AFHQ classifier classifying class 'cat' (ablation with 3 symbols in the final codebook of the hierarchy), with atoms corresponding to $pf(cat, \zeta_0^2)$ described in the first row. The second row visualises the obtained atoms corresponding to $pf(\zeta_0^2, \zeta_{21}^1)$. The last row visualises the atoms corresponding to symbols in ζ^0 which are part of ζ_{21}^1.

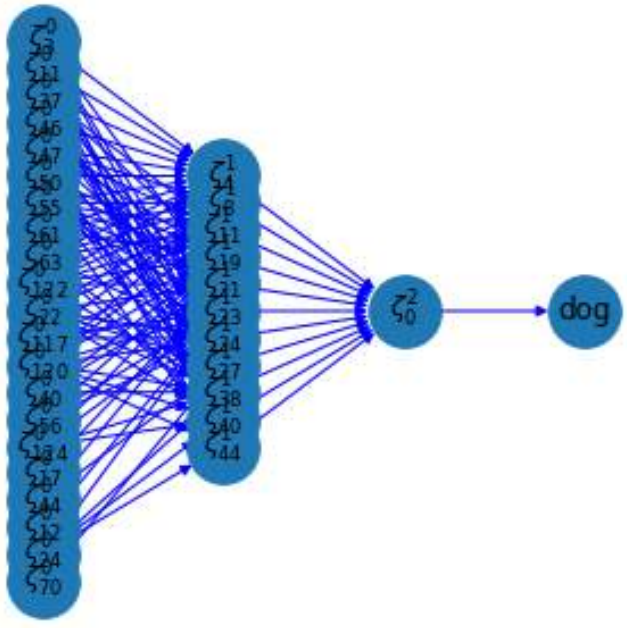
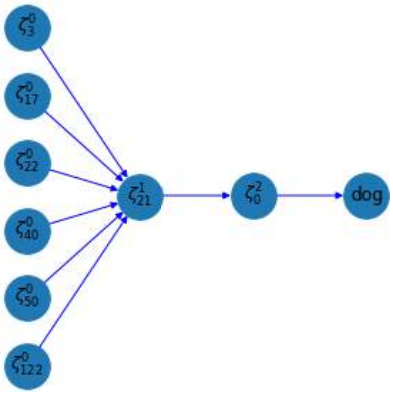
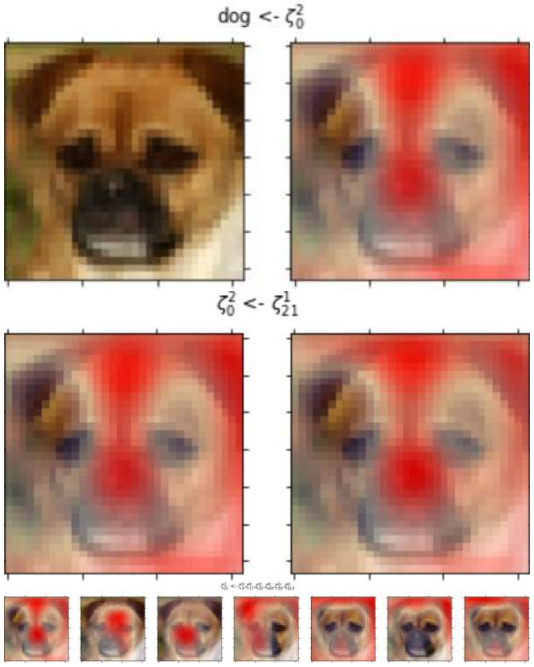


Figure 8. This figure shows the class-level tree for 'dog' class

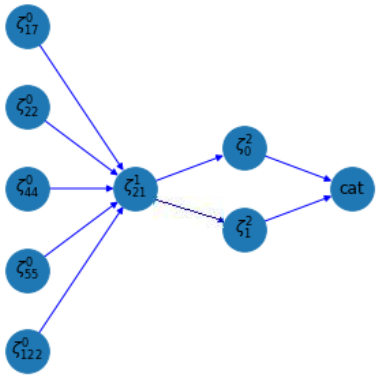


(a) Image-level tree

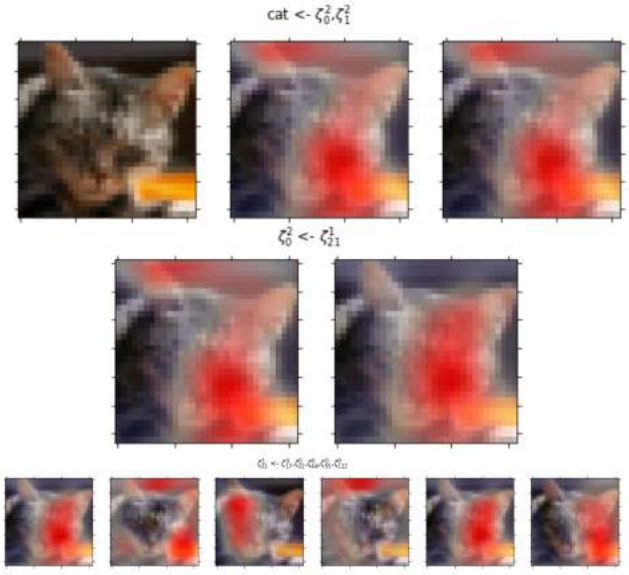


(b) Visual rules

Figure 9. This figure describes the hierarchical visual explanations obtained using our proposed framework for the AFHQ classifier for class dog, with atoms corresponding to $pf(dog, \zeta_0^2)$ described in the first row. The second row visualises the obtained atoms corresponding to $pf(\zeta_0^2, \zeta_{21}^1)$. The last row visualises the atoms corresponding to symbols in ζ^0 which are part of ζ_{21}^1.

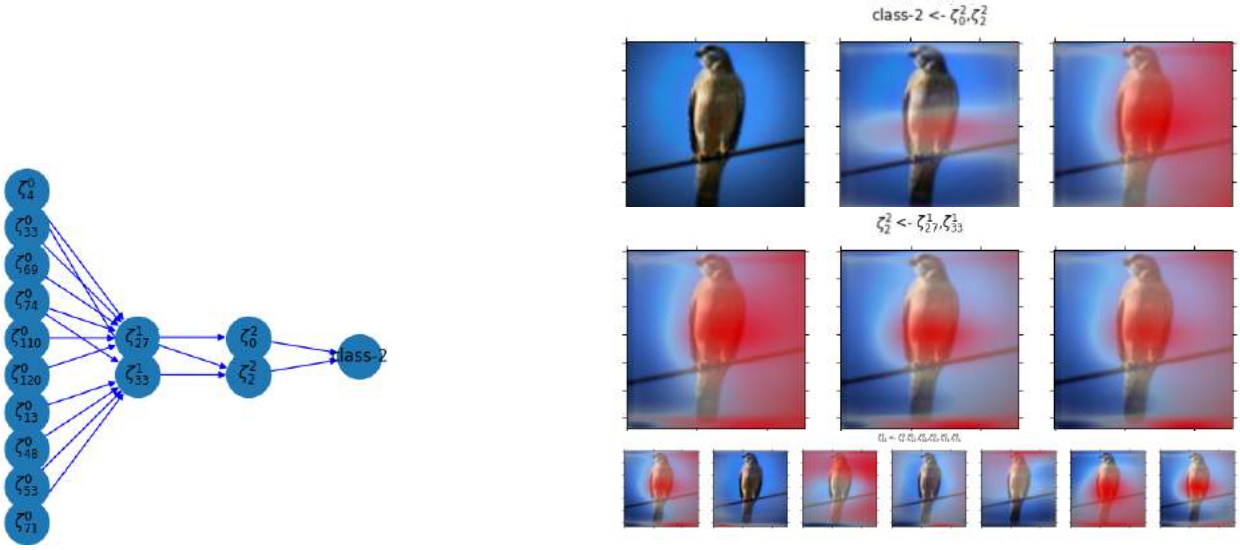


(a) Image-level tree



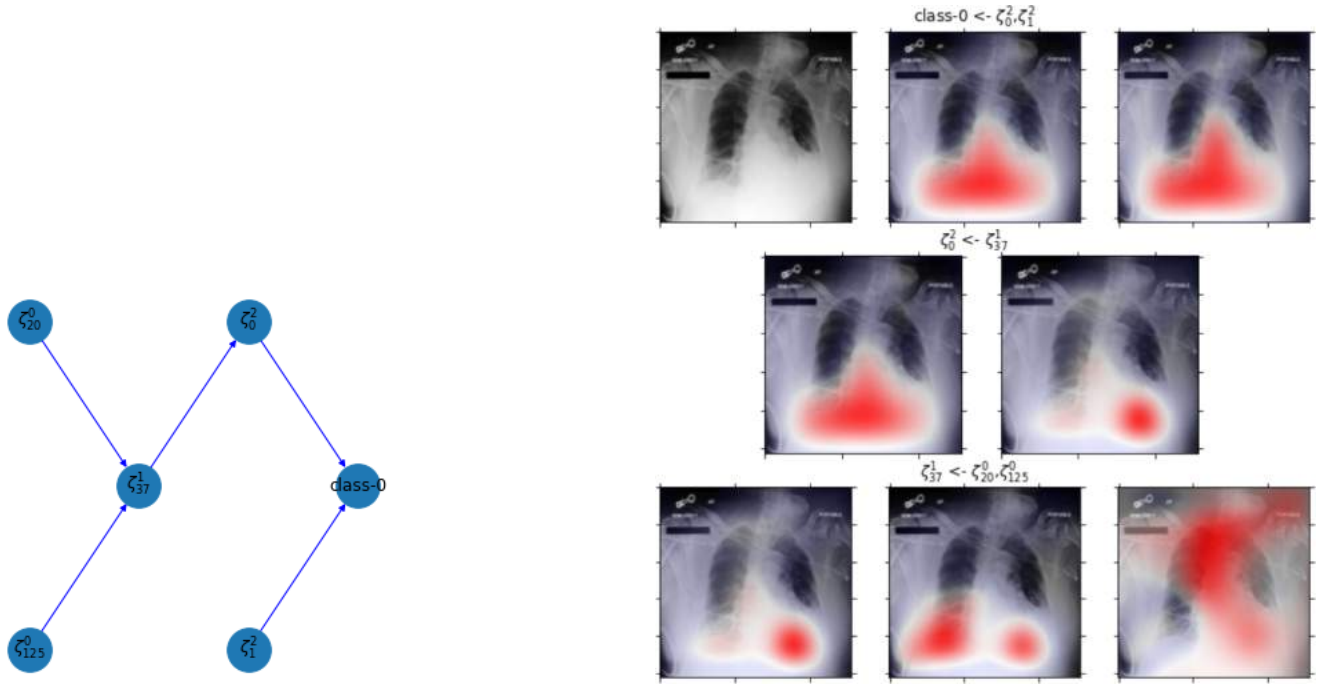
(b) Visual rules

Figure 10. This figure describes the hierarchical visual explanations obtained using our proposed framework for the AFHQ classifier classifying class 'cat', with atoms corresponding to $pf(cat, \zeta_0^2), pf(cat, \zeta_1^2)$ described in the first row. The second row visualises the obtained atoms corresponding to $pf(\zeta_0^2, \zeta_{21}^1)$. The last row visualises the atoms corresponding to symbols in ζ^0 which are part of ζ_{21}^1.



(a) Image-level tree (b) Visual rules

Figure 11. This figure describes the hierarchical visual explanations obtained using our proposed framework for the STL10 classifier classifying class 'bird(class 2)' , with atoms corresponding to $pf(bird, \zeta_0^2), pf(bird, \zeta_2^2)$ described in the first row. The second row visualises the obtained atoms corresponding to $pf(\zeta_2^2, \zeta_{27}^1), pf(\zeta_2^2, \zeta_{33}^1)$. The last row visualises the atoms corresponding to symbols in ζ^0 which are part of ζ_{33}^1.



(a) Image-level tree (b) Visual rules

Figure 12. This figure describes the hierarchical visual explanations obtained using our proposed framework for the MIMIC classifier classifying class 'abnormal(class 0)' , with atoms corresponding to $pf(class0, \zeta_0^2), pf(class0, \zeta_1^2)$ described in the first row. The second row visualises the obtained atoms corresponding to $pf(\zeta_0^2, \zeta_{37}^1)$. The last row visualises the atoms corresponding to symbols in ζ^0 which are part of ζ_{37}^1.

References

[1] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019. 1, 2

[2] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in neural information processing systems*, 31, 2018. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 2

[4] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. *Advances in neural information processing systems*, 32, 2019. 2

[5] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014. 2

[6] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 4

[7] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 4

[8] 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 *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 4

[9] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017. 4

[10] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR, 2020. 3

[11] Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. *Advances in Neural Information Processing Systems*, 29, 2016. 2