HYPERPG - PROTOTYPICAL GAUSSIANS ON THE HYPERSPHERE FOR INTERPRETABLE DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Prototype Learning methods provide an interpretable alternative to black-box deep learning models. Approaches such as ProtoPNet learn, which part of a test image "look like" known prototypical parts from training images, combining predictive power with the inherent interpretability of case-based reasoning. However, existing approaches have two main drawbacks: A) They rely solely on deterministic similarity scores without statistical confidence. B) The prototypes are learned in a black-box manner without human input. This work introduces HyperPg, a new prototype representation leveraging Gaussian distributions on a hypersphere in latent space, with learnable mean and variance. HyperPg prototypes adapt to the spread of clusters in the latent space and output likelihood scores. The new architecture, HyperPgNet, leverages HyperPg to learn prototypes aligned with human concepts from pixel-level annotations. Consequently, each prototype represents a specific concept such as color, image texture, or part of the image subject. A concept extraction pipeline built on foundation models provides pixel-level annotations, significantly reducing human labeling effort. Experiments on CUB-200-2011 and Stanford Cars datasets demonstrate that HyperPgNet outperforms other prototype learning architectures while using fewer parameters and training steps. Additionally, the concept-aligned HyperPg prototypes are learned transparently, enhancing model interpretability.

1 INTRODUCTION

Deep Learning has achieved high accuracy in many computer vision tasks. However, the decision-making processes of these models lack transparency and interpretability, making deployment in safety-critical areas challenging. Explainable Artificial Intelligence (XAI) seeks to develop interpretability methods to open the black-box reasoning processes of these models and increase trust in their decisions.

XAI methods can be broadly divided into two categories: First, Post-Hoc Methods like LIME (Ribeiro et al., 2016), SHAP (Lundberg & Lee, 2017) or GradCAM Selvaraju et al. (2017) offer explanations for predictions without requiring retraining. While applicable in many scenarios, post-hoc methods may not actually align with the models' decision making processes, potentially leading to interpretations that are not entirely faithful (Rudin, 2019). Second, inherently interpretable methods provide built-in, case-based reasoning processes. For instance, small decision trees are inherently interpretable because their reasoning can be easily understood as a series of if-else statements (Molnar, 2020). However, they are constrained in their representational power.

Deep Prototype Learning Architectures such as ProtoPNet (Chen et al., 2019) and its derivatives (e.g., Rymarczyk et al., 2020; Donnelly et al., 2021; Sacha et al., 2023) integrate inherent interpretability into deep learning models through a prototype layer. Each neuron in this layer represents a prototype, storing a latent feature vector. The model's predictions are based on the distances between sample features and prototype parameters, for example by computing the L_2-distance. However, these deterministic similarity scores do not include statistical information like confidence. To include such contextual information, prototypes could be modeled as probability distributions, like Gaussian distributions. Yet, recent prototype architectures favor the cosine hypersphere as feature space for its classification advantages (Mettes et al., 2019), where defining a Gaussian distribution analogue is challenging (Hillen et al., 2017).

The prototypes in ProtoPNet and its successors are learned in an opaque manner via backpropagation. After training, the model "pushes" the learned prototypes on the embedding of the most similar known training sample (Chen et al., 2019). However, these prototypes are learned without human control and do not incorporate domain knowledge. A new training regime is needed to aligned prototypes with human-defined concepts. The main contributions of this paper are as follows:

- **HyperPg**: A new prototype representation with learned parameters anchor α, mean μ and standard deviation σ. This representation models a Gaussian distribution over cosine similarities, thereby projecting a Gaussian distribution on the surface of a hypersphere. HyperPg's similarity score is based on the Gaussian's probability density function and adapts its size through a learned standard deviation.
- **HyperPgNet**: A new prototype learning architecture built on HyperPg prototypes. HyperPgNet learns prototypes aligned with human-defined concepts using pixel level annotations. These concepts describe features such as color and patterns of birds or car parts.
- **Concept Extraction Pipeline**: Based on Grounding DINO (Liu et al., 2023) and SAM2 (Ravi et al., 2024), this pipeline provides pixel-level concept annotations at scale.
- **Classification Experiments**: HyperPgNet is compared with other prototype learning architectures like ProtoPNet on the CUB-200-2011 (Wah et al., 2011) and Stanford Cars (Krause et al., 2013) datasets. HyperPgNet outperforms the other models with fewer learned prototypes, while infusing each prototype with more meaning.

2 Related Work

Prototype Learning. In image classification, prototype learning approaches using autoencoders provide high interpretability by reconstructing learned prototypes from latent space back to the image space (Li et al., 2018). However, these approaches are limited in their performance because each prototype must represent the entire image. Alternatively, parts-based approaches like ProtoPNet (Chen et al., 2019; Donnelly et al., 2021) learn a predefined number of prototypical parts per class. Each class-specific prototype is an image patch represented by a 1×1 spatial unit in the latent space. The network classifies images by summarizing the similarities of all the latent image patches to all prototypes using a fully connected layer. PIPNet (Nauta et al., 2023) aligns learned prototypes with human perception by optimizing the model to assign the same prototype to two different views of the same augmented image patch. ProtoPShare (Rymarczyk et al., 2020) merges similar prototypes after training is complete, while ProtoPool (Rymarczyk et al., 2022) directly learns class-shared prototypes. Other approaches additionally replace the linear output layer with other interpretable models. ProtoKNN (Ukai et al., 2023) classifies the similarity scores using a k-nearest neighbor (KNN) and ProtoTree Nauta et al. (2021) learn a decision tree.

HyperPgNet learns a predefined number of prototypes per concept, not per class. These concepts are provided as pixel-level annotations extracted by a new labeling pipeline based on the foundation models DINOv2 (Oquab et al., 2024) and SAM2 (Ravi et al., 2024). Similar to part-based prototypes, the concept prototypes span a 1×1 spatial unit in the latent space. The prototypes are exclusive to each concept but shared among the classes.

Prototype learning is also applied to image segmentation. ProtoSeg (Sacha et al., 2023) uses prototypical parts a latent space with high spatial resolution for image segmentation. Another method proposes non-learnable prototypes (Zhou et al., 2022), which are trained by clustering in the latent space instead of being optimized via Backpropagation. ProtoGMM (Moradinasab et al., 2024) builds on this by updating class-specific prototypes like a Gaussian Mixture Model (GMM) using an Expectation-Maximization (EM) algorithm (Liang et al., 2022) with fixed mixture weights. MGProto (Wang et al., 2023) also learns mixture weights for their prototype GMM model.

HyperPgNet employs HyperPg, a novel prototype representation using Gaussian distributions on a hypersphere's surface with learned parameters: direction $\boldsymbol{\alpha}$, mean μ and standard deviation (std) σ. Unlike ProtoGMM, which learns a multivariate Gaussian distribution in Euclidian space, each HyperPg prototype is defined by a directional vector $\boldsymbol{\alpha}$ and a $1D$-Gaussian distribution of cosine similarity values with mean μ and standard-deviation σ around this vector. The use of cosine similarity for prototype learning has been shown to improve classification (Mettes et al., 2019) and has been widely adopted. (e.g., Donnelly et al., 2021; Rymarczyk et al., 2022; Sacha et al., 2023).

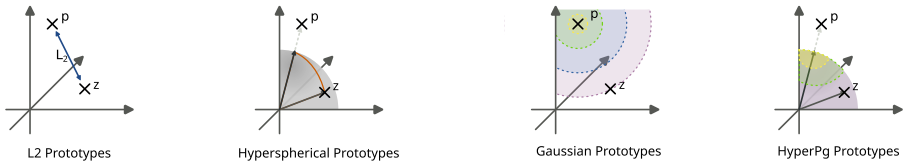


Figure 1: Illustration on how the different prototype formulations compute the similarity between a prototype $\boldsymbol{p}$ and latent vector $\boldsymbol{z}$. L_2 prototypes compute the Euclidian distance between two points in latent space. Hyperspherical prototypes use the cosine similarity of normalized vectors, which corresponds to the angle between two points on a hypersphere. Gaussian prototypes model a Gaussian distribution in Euclidian space and compute a probability density function (PDF). HyperPg prototypes learn a Gaussian distribution of cosine similarities, thereby projecting a Gaussian distribution onto the surface of a hypersphere.

HyperPgNet advances this by computing prototype activations as the probability density of each HyperPg prototype's Gaussian distribution, allowing each prototype to adapt to the spread of the latent cluster.

Concept-Based Methods. Testing with Concept Activation Vectors (TCAV) (Kim et al., 2018) is a post-hoc method that aligns model activations with human-understandable concepts. In image classification, TCAV measures model activations on images representing a specific concept like "striped" and compares them to activations on a random image set. Concept Activation Regions (CAR) (Crabbé & van der Schaar, 2022) relax TCAV's assumption of linear separability in latent space by using support vector machines and the kernel trick. Another post-hoc approach, CRAFT (Fel et al., 2023), generates concept attribution maps by identifying relevant concepts from the training set via Non-Negative Matrix Factorization (NMF) and backpropagating with GradCAM (Selvaraju et al., 2017).

Instead of analyzing the learned concepts post-hoc, some approaches integrate concept learning into the classifier. MCPNet (Wang et al., 2024) extracts concepts from multiple layers by splitting feature maps along the channel dimension, learning concepts during training and at multiple scales. Concept Bottleneck Models (CBMs) (Koh et al., 2020) augment black-box models to predict predefined concept labels in an intermediate step. In this manner, CBMs behave like two sequential black box models and their internal reasoning remains opaque.

HyperPgNet combines the concept-aligned learning approach of CBM with interpretable prototype learning. HyperPgNet learns multiple HyperPg prototypes per concept, instead of a fully connected concept prediction layer. A novel concept extraction pipeline, built on foundation models, provides pixel level annotations at scale. By using those annotations to learn concept prototypes, HyperPgNet avoids the ambiguity of aligning post-hoc explanations with the model's internal reasoning.

3 Prototypical Gaussians on the Hypersphere

Prototype Learning is an inherently interpretable machine learning method. The reasoning process is based on the similarity scores of the inputs to the prototypes, retained representations of the training data. For example, a K-Nearest Neighbor (KNN) model is a prototype learning approach with the identity function for representation and an unlimited number of prototypes. In contrast, a Gaussian Mixture Model (GMM) uses a mean representation but restricts the number of prototypes to the number of mixture components.

Prototype learning for deep neural networks involves finding structures in latent space representations. This section provides an overview of existing methods, which are also illustrated in Fig. 1. Prior work uses point-based prototypes, computing similarity scores relative to a single point in latent space. Aligning prototypes with human-labeled concepts requires a more powerful prototype representation, leading to the introduction of HyperPg - Prototypical Gaussians on the Hypersphere. HyperPg prototypes are able to adapt their shape in the latent space by modeling Gaussian distributions with mean and standard deviation, adapting to the variance in encoded concept features.

3.1 POINT BASED PROTOTYPES

The general formulation of prototypes, as defined in previous work (e.g., Chen et al., 2019), is discussed first. Let $\mathcal{D} = [\boldsymbol{X}, \boldsymbol{Y}] = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ denote the training set, e.g., a set of labeled images, with classes C. Each class $c \in C$ is represented by Q many prototypes $\boldsymbol{P}_c = \{\boldsymbol{p}_{c,j}\}_{j=1}^{Q}$.

Some feature encoder Enc projects the inputs into a D-dimensional latent space $\mathcal{Z}$, with $\boldsymbol{z}_i = \mathrm{Enc}(\boldsymbol{x}_i)$ being a feature map of shape $\zeta_w \times \zeta_h \times D$ with spatial size $\zeta = \zeta_w \zeta_h$. Commonly, the prototypes $\boldsymbol{p}$ are also part of $\mathcal{Z}$ with shape $\rho_w \times \rho_h \times D$, i.e., spatial size $\rho = \rho_w \rho_h$.

Autoencoder approaches use $\rho = \zeta$, meaning the prototype represents the entire image and can be reconstructed from latent space (Li et al., 2018). Part-based approaches like ProtoPNet and segmentation models like ProtoSeg use $\rho = 1$ (Chen et al., 2019; Zhou et al., 2022), meaning each prototype represents some part of the image. Notable exceptions include Deformable ProtoPNet (Donnelly et al., 2021), where each prototype has spatial size $\rho = 3 \times 3$ and MCPNet (Wang et al., 2024), where prototypes are obtained by dividing the channels of the latent space into chunks.

The prediction is computed by comparing each prototype $\boldsymbol{p}$ to the latent feature map $\boldsymbol{z}$. For simplicity's sake lets assume the spatial dimensions $\rho = \zeta = 1$. The following equations can be adapted for higher spatial dimensions by summing over $\sum_{\rho_w} \sum_{\rho_h}$ for each chunk of the latent map.

ProtoPNet's prototypes leverage the L_2 similarity. The L_2 similarity measure is defined as

$$s_{L_2}(\boldsymbol{z}|\boldsymbol{p}) = \log\left(\frac{\|\boldsymbol{z} - \boldsymbol{p}\|_2^2 + 1}{\|\boldsymbol{z} - \boldsymbol{p}\|_2^2 + \epsilon}\right) \tag{1}$$

and is based on the inverted L_2 distance between a latent vector $\boldsymbol{z}$ and a prototype vector $\boldsymbol{p}$. This similarity is a point-based measure, as only the two vectors are compared, without any additional context like the expected variance of the cluster represented by the prototype.

Hyperspherical prototypes using the cosine similarity have been shown to perform well in classification tasks (Mettes et al., 2019) and have been widely used since (e.g., Zhou et al., 2022; Ukai et al., 2023). The cosine similarity is defined as

$$s_{\cos}(\boldsymbol{z}|\boldsymbol{p}) = \frac{\boldsymbol{z}^\top \boldsymbol{p}}{\|\boldsymbol{z}\|_2 \|\boldsymbol{p}\|_2}, \tag{2}$$

which is based on the angle between two normalized vectors of unit length. By normalizing D dimensional vectors to unit length, they are projected onto the surface of a D dimensional hypersphere. The cosine similarity is defined on the interval $[-1, 1]$ and measures: 1 for two vectors pointing in the same direction, 0 for orthogonal vectors, and -1 for vectors pointing in opposite directions. Like the L_2 similarity, the cosine similarity is a point-based measure comparing only two vectors.

Both the L_2 and cosine similarity can be used for classification. The similarity scores are processed by a fully connected layer (e.g., Chen et al., 2019; Donnelly et al., 2021), or a winner-takes-all approach assigns the class of the most similar prototype (e.g., Sacha et al., 2023). Prototypes can be learned by optimizing a task-specific loss, such as cross-entropy, via backpropagation. Alternatively, Zhou et al. (2022) propose "non-learnable" prototypes, whose parameters are obtained via clustering in the latent space rather than backpropagation.

3.2 GAUSSIAN PROTOTYPES

Gaussian prototypes model prototypes as a Gaussian distribution with mean and covariance. They adapt to the spread of the associated latent cluster by adjusting their covariance matrix. Thus, a Gaussian prototype with a wide covariance can still have a relatively high response even for larger distances from the mean vector.

Let the formal definition of a Gaussian prototype be $\boldsymbol{p}^G = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The parameters of $\boldsymbol{p}_{c,j}^G$ now track both the mean and covariance of latent vector distribution $\boldsymbol{Z}_{c,j}$. Each Gaussian prototype $\boldsymbol{p}^G$ thus defines a multivariate Gaussian Distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Gaussian prototypes can be trained using EM for clustering in the latent space (Zhou et al., 2022; Wang et al., 2023; Moradinasab et al., 2024) or by directly optimizing the parameters via Backpropagation. The similarity measure of Gaussian prototypes is defined as the probability density function (PDF) for D-dimensional multivariate

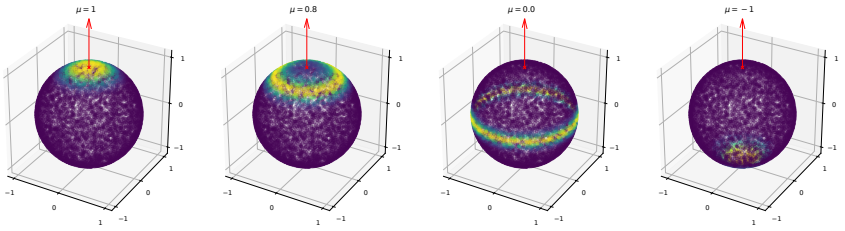


Figure 2: HyperPg learns a Gaussian distribution with mean μ and std σ of cosine similarities with the anchor $\boldsymbol{\alpha}$. The plots visualize HyperPg's activations on the $3D$ hypersphere with anchor $\boldsymbol{\alpha} = (0, 0, 1)$, std $\sigma = 0.1$ and and various mean cosine similarities $\mu \in [-1, 1]$ for 10k random samples. The anchor $\boldsymbol{\alpha}$ is shown as a red arrow, and std σ governs the width of the distribution. For $\mu = 1$, the distribution is aligned with the vector; for $\mu = -1$ it is on the opposite side of the hypersphere. Due to the cosine similarity, setting $1 > \mu > -1$ interpolates the area of highest probability density between both poles, leading to a ring shape on the hypersphere's surface.

Gaussians, namely

$$s_{\text{Gauss}}(\boldsymbol{z}|\boldsymbol{p}^G) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}) \tag{3}$$

This formulation as a PDF has several advantages: A) The similarity can be interpreted as the likelihood of being sampled from the Gaussian prototypes, which is more meaningful than a distance metric in a high-dimensional latent space. B) Prototypes can adapt their shape using a full covariance matrix, allowing different variances along various feature dimensions, offering more flexibility in shaping the latent space. However, this increases computational requirements, especially with EM clustering. Gaussians prototypes lose the advantages of hyperspherical prototypes for classification and regression Mettes et al. (2019).

3.3 Gaussian Prototypes on the Hypersphere - HyperPg

Prototypical Gaussians on the Hypersphere (HyperPg) combine the advantages of Gaussian and hyperspherical prototypes. HyperPg prototypes are defined as $\boldsymbol{p}^H = (\boldsymbol{\alpha}, \mu, \sigma)$ with a directional anchor vector $\boldsymbol{\alpha}$, *scalar* mean similarity μ and *scalar* standard deviation (std) σ. HyperPg prototypes learn a $1D$ Gaussian distribution over the cosine similarities to the anchor vector $\boldsymbol{\alpha}$. Because the cosine similarity is bounded to $[-1, 1]$, HyperPg's similarity measure is defined as the PDF of the truncated Gaussian distribution within these bounds. Let $\mathcal{G}(x, \mu, \sigma)$ be the cumulative Gaussian distribution function. Then, HyperPg's similarity measure based on the truncated Gaussian distribution is defined as

$$s_{\text{HyperPg}}(\boldsymbol{z}|\boldsymbol{p}^H) = \mathcal{T}_G(s_{\cos}(\boldsymbol{z}|\boldsymbol{\alpha}); \mu, \sigma, -1, 1) \tag{4}$$

$$= \frac{\mathcal{N}\left(s_{\cos}(\boldsymbol{z}|\boldsymbol{\alpha}); \mu, \sigma\right)}{\mathcal{G}(1, \mu, \sigma) - \mathcal{G}(-1, \mu, \sigma).} \tag{5}$$

Fig. 2 illustrates the activations of HyperPg's similarity function on the surface of a $3D$ hypersphere with anchor $\boldsymbol{\alpha} = (0, 0, 1)$, fixed std $\sigma = 0.1$ and various mean values $\mu \in [-1, 1]$. The anchor $\boldsymbol{\alpha}$ defines a prototypical direction vector in latent space $\mathcal{Z}$, similar to other hyperspherical prototypes, and is visualized as a red arrow. The learned Gaussian distribution of cosine similarities is projected onto the hypersphere's surface with std σ governing the spread of the distribution, and mean μ the expected distance to the anchor $\boldsymbol{\alpha}$. For $\mu = 1$, the distribution centers around the anchor, as the cosine similarity is 1 if two vectors point in the same direction. For $\mu = -1$, the distribution is the on opposite side of the hypersphere, as the cosine similarity is -1 for vectors pointing in opposite directions.

For values of $1 > \mu > -1$, the distribution forms a hollow ring around the anchor vector $\boldsymbol{\alpha}$. This occurs because the cosine similarity for these μ values expects the activating vectors to point in a different direction than the anchor, without specifying the direction. Imagining the interpolation between $\mu = 1$ and $\mu = -1$, the probability mass moves from one pole of the hypersphere to the other, stretching like a rubber band over the surface. For $\mu = 0$, the expected cosine similarity

indicates that vectors with the highest activation are orthogonal to the anchor $\boldsymbol{\alpha}$. Since no specific direction is indicated, the entire hyperplanar segment orthogonal to the anchor has the highest activation. This activation pattern for the cosine similarity would typically require an infinite mixture of prototype vectors pointing in all directions in this hyperplane. HyperPg achieves the same effect by learning only one prototype vector (the anchor) and just two additional scalar parameters. This significantly increases HyperPg's representational power compared to standard hyperspherical and Gaussian prototypes, as a single HyperPg prototype can cover multiple clusters simultaneously.

Combining the strengths of hyperspherical and Gaussian prototypes, HyperPg can learn more complex structures in the latent space, such as human-defined concepts, and provide a more meaningful similarity measure. HyperPg can be easily adapted to other probability distributions with additional desirable properties. Possible candidate distributions are elaborated on in Appendix A. Similarly, it is possible to exchange the cosine similarity to other similarity measures or functions, and learn an untruncated PDF over their output, making the HyperPg idea transferable to applications outside of prototype learning.

4 TRAINING

The original ProtoPNet implementation uses three loss functions: a task specific loss like crossentropy for classification, a cluster loss to increase compactness within a class's cluster, and a separation loss to increase distances between different prototype clusters. However, the prototypes of ProtoPNet and its successors are learned in a black-box manner.

HyperPgNet is a new inherently interpretable deep learning approach built on HyperPg prototypes. It introduces a "Right for the Right Concept" loss, inspired by "Right for the Right Reasons" (Ross et al., 2017), to restrict the learned prototypes to human-defined concepts. This focus enhances the interpretability and minimizes the influence of confounding factors. This section first provides an overview of the used prototype learning losses, then introduces the Right for the Right Concept loss, and finally discusses the overall network architecture and final multi-objective loss.

4.1 PROTOTYPE LOSSES

ProtoPNet defines a cluster loss function to shape the latent space such that all latent vectors $\boldsymbol{z}_c \in \boldsymbol{Z}_c$ with class label c are clustered tightly around the semantically similar prototypes $\boldsymbol{p}_c \in \boldsymbol{P}_c$. The cluster loss function is defined as

$$L_{\text{Clst}} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C|}\sum_{c\in C}\max_{\boldsymbol{p}_c\in\boldsymbol{P}_c}\max_{\boldsymbol{z}_{c,i}\in\boldsymbol{Z}_{c,i}} s(\boldsymbol{p}_c, \boldsymbol{z}_{c,i}), \tag{6}$$

where $s(\cdot,\cdot)$ is some similarity measure. The L_{Clst}-Loss function increases compactness by increasing the similarity between prototypes $\boldsymbol{p}_c$ of class c latent embeddings $\boldsymbol{z}_c$ of class c over all samples.

An additional separation loss increases the margin between different prototypes. The separation loss function is defined as

$$L_{\text{Sep}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C|}\sum_{c\in C}\max_{\boldsymbol{p}_{\neg c}\notin\boldsymbol{P}_c}\max_{\boldsymbol{z}_{c,i}\in\boldsymbol{Z}_{c,i}} s(\boldsymbol{p}_{\neg c}, \boldsymbol{z}_{c,i}), \tag{7}$$

The L_{Sep} function punishes high similarity values between a latent vector $\boldsymbol{z}_c$ of class c and prototypes $\boldsymbol{p}_{\neg c}$ not belonging to c, thereby separating the clusters in latent space. Please note, ProtoPNet (Chen et al., 2019) use a slightly different notation by working with the L_2 *distance*, instead of a *similarity* measure.

In HyperPgNet the learned prototypes are not class exclusive, but shared among different classes. Instead, each prototype is assigned to one human-defined concept $k \in K$. The Concept Activation

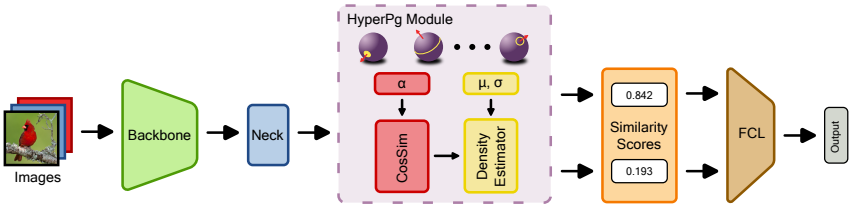


Figure 3: HyperPgNet Architecture. The HyperPg module can be easily exchanged to other prototype formulations such as ProtoPNet. HyperPgNet uses the truncated Gaussian distribution as density estimator, but other PDFs are possible.

Regions (CAR, Crabbé & van der Schaar, 2022) model proposes a concept density loss defined as

$$L_{Density} = -\frac{1}{|K|}\sum_{k\in K}\frac{1}{|\boldsymbol{P}_k|}\sum_{\boldsymbol{p}_k\in\boldsymbol{P}_k}\phi(\boldsymbol{p}_k,\boldsymbol{Z}_k)-\phi(\boldsymbol{p}_k,\boldsymbol{Z}_{\neg k}) \tag{8}$$

$$\phi(\boldsymbol{p},\boldsymbol{Z}) = \frac{1}{|\boldsymbol{Z}|}\sum_{\boldsymbol{z}\in\boldsymbol{Z}} s(\boldsymbol{p},\boldsymbol{z}). \tag{9}$$

The density loss uses the similarity aggregation $\phi(\boldsymbol{p},\boldsymbol{Z})$ which computes the mean response of a prototype $\boldsymbol{p}$ with a set of latent features $\boldsymbol{Z}$. The density loss therefore computes the mean response of correctly assigned prototypes minus the mean response of incorrectly assigned prototypes. The loss functions proposed by ProtoPNet compute the *maximum* response of correctly and incorrectly assigned prototypes instead.

4.2 Right Concept Loss

To ensure prototypes actually correspond to the input pixels containing the concept, and do not respond to other factors in the background, HyperPgNet introduces the "Right for the Right Concept" (RRC) -Loss inspired by "Right for the Right Reasons" (RRR, Ross et al., 2017). The RRC loss is defined as

$$L_{\mathrm{RRC}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k\in K}\left(\boldsymbol{A}_{\boldsymbol{x}_i,k}\frac{\partial}{\partial\boldsymbol{x}_i}\sum_{\boldsymbol{p}_k\in\boldsymbol{P}_k} s\left(\boldsymbol{p}_k,\mathrm{Enc}\left(\boldsymbol{x}_i\right)\right)\right)^2, \tag{10}$$

with binary annotation matrix $\boldsymbol{A}_{\boldsymbol{x}_i,k} \in \{0,1\}^{N\times\times W\times H}$ for each input sample $\boldsymbol{x}_i$ and concept k. This annotation matrix defines for each input image, which pixels contain which concept. In the original RRR paper, the annotation matrix specified relevant regions for the classification task, steering the model's activations away from confounding factors in the background. In HyperPgNet the RRC-Loss further strengthens the prototype-concept association.

4.3 Multi-Objective Loss Function

To train a prototype learning network like HyperPgNet for downstream tasks like image classification, a multi-objective loss function is employed. This multi-objective loss function is defined as

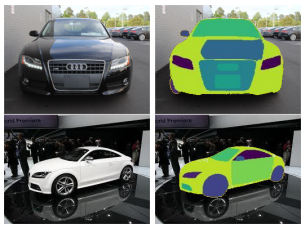
$$L = L_{\mathrm{CE}} + \lambda_{\mathrm{Clst}}L_{\mathrm{Clst}} + \lambda_{\mathrm{Sep}}L_{\mathrm{Sep}} + \lambda_{\mathrm{RRC}}L_{\mathrm{RRC}},$$

where L_{CE} is the cross-entropy loss over network predictions and ground truth image labels. ProtoPNet uses $\lambda_{\mathrm{Clst}} = 0.8$ and $\lambda_{\mathrm{Sep}} = 0.08$ (Chen et al., 2019). For HyperPgNet, the different loss terms are weighted equally, meaning all $\lambda = 1$.

4.4 Network Architecture

Fig. 3 illustrates HyperPgNet's Architecture for interpretable image classification. HyperPgNet uses a pretrained, off-the-shelf feature encoder as a backbone, such as the convolution pyramid from a CNN like ConvNeXt-tiny (Liu et al., 2022) or a transformer architecture like Segformer (Xie et al., 2021). The high-dimensional feature maps provided by the backbone are passed through a

(a) Concept Segmentation Masks for CUB-200-2011 based on provided positive and negative part annotations.



(b) Concept Segmentation Mask for Stanford Cars generated from a list of human-defined car parts.

Figure 4: Examples for Automated Concept Extraction.

neck, consisting of two 1×1 convolution layers with ReLU activation in between, projecting into a smaller latent space with 256 channels, followed by Layer Normalization. The hyperparameter of 256 channels aligns with previous work (e.g., Rymarczyk et al., 2022).

HyperPg's similarity function computes the truncated Gaussian PDF over the cosine similarity of the latent feature maps with the HyperPg prototype anchors $\boldsymbol{\alpha}$. HyperPgNet implements the prototype learning part of the network as a module with two layers: First, a prototype learning layer computes the cosine similarity of learnable HyperPg prototype anchors α to the latent feature map z, similar to other prototype learning architectures. Second, the new Density Estimation layer with learnable parameters mean μ and std σ computes the truncated Gaussian PDF over the activations of the previous layer.

This implementation of the HyperPg Module with two internal layers supports swapping out the prototype similarity measure to other measures in the future. Additionally, the Density Estimation Layer concept could be applied to other architectures to learn a Gaussian distribution over the previous layer's activations. Finally, the output of the HyperPg Module is passed through a single fully connected layer (FCL) to produce the output logits or class scores.

5 Concept Extraction at Scale

To learn prototypes based on human-defined concepts with RRC-Loss for image classification, pixel level annotations are required. Foundation models are leveraged to generate these annotations, reducing the need for human labeling.

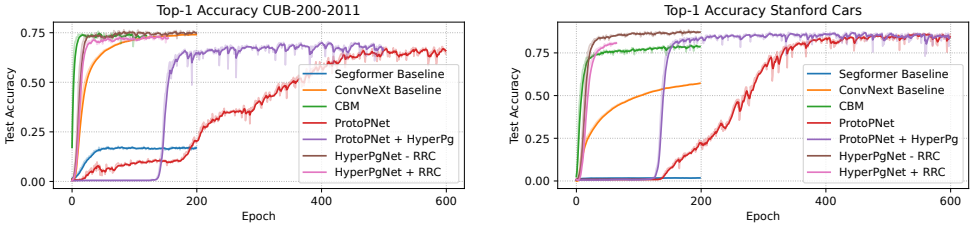
The CUB-200-2011 dataset (Wah et al., 2011) contains 11,788 images of birds across 200 subspecies. Each image is annotated with up to 15 part locations, such as head, tail and back. The dataset includes 312 binary attributes describing patterns and colors of these parts, or additional shape information like "duck-like" or "owl-like". For concept learning, 30 attributes are manually selected, excluding small concepts like eye-color or class exclusive attributes like "duck-like".

To extract pixel level annotation masks, the point annotations for the attributes are given to a Segment Anything 2 (SAM2) model(Ravi et al., 2024). Using the point locations belonging to other attributes as negative points, segmentation masks are generated (see Fig. 4a). The segmentation mask is then assigned the attribute from the dataset annotations to define a concept.

The Stanford Cars dataset (Krause et al., 2013) does not provide part annotations. This dataset contains 16,185 images of 196 car classes, categorized by make, model and year (e.g., BMW M3 coupe 2012). The dataset provides only bounding boxes for the main subject matter, useful for foreground-background separation. To obtain pixel-level annotations, a list of 10 car parts (e.g., "wheel", "headlight", "radiator") is defined. A Grounding DINO model (Liu et al., 2023) uses this list to generate bounding boxes for parts in the image. A post-processing step removes possible outliers by eliminating bounding boxes outside the original bounding box provided by the dataset.Finally, SAM2 generates pixel-level segmentation masks (see Fig. 4b). For Stanford Cars, only the part label is used as the concept label as the dataset does not provide additional part attributes. The automated pipeline labeled the entire Stanford Cars dataset within 2h on an NVidia 4060ti with 16GB VRAM and can be easily adapted to other datasets.

Table 1: Test Top-1 Accuracy with Segformer Backbone. Models are grouped by their reasoning process. BB: Black Box. PP: Prototypical Parts. CAP: Concept Aligned Prototypes.

		CUB-200-2011		Stanford Cars	
		# Prototypes	**Accuracy [%]**	**# Prototypes**	**Accuracy [%]**
BB	Segformer Baseline	-	17.7	-	1.9
	ConvNeXt Baseline	-	74.2	-	57.5
	CBM	-	75.7	-	79.9
PP	ProtoPNet	2000	68.0	1960	86.4
	ProtoPNet + HyperPg	2000	70.5	1960	87.4
CAP	**HyperPgNet - L_{RRC}**	**300**	**76.5**	**180**	**88.6**
	HyperPgNet + L_{RRC}	**300**	**74.1**	**180**	**81.2**



(a) CUB-200-2011 (b) Stanford Cars

Figure 5: Test Accuracy with Segformer Backbone.

6 Experiments

The experiments were performed on two datasets: The CUB-200-2011 bird species classification dataset (Wah et al., 2011) and the Stanford Cars dataset (Krause et al., 2013). Two variants of HyperPgNet were tested, with and without the L_{RRC} loss. HyperPgNet was compared to ProtoPNet with both standard L_2 and the newly introduced HyperPg prototypes. Concept Bottleneck Model (CBM) with joint training scheme (Koh et al., 2020) is a black box model which also uses the concept annotations. Segformer MiT-B4 and ConvNext served as Transformer and CNN baselines, respectively. All models were implemented with pretrained Segformer MiT-B4 backbones in Pytorch. However, HyperPgNet's performance is independent of the chosen backbone architecture (see Appendix B.1).

In contrast to prior work, the training and testing was done on full images, without crops to the bounding boxes provided by the datasets. For data augmentation RandomPerspective, RandomHorizontalFlip, and RandomAffine were used in an online manner. Thus, for CUB-200-2011, each class has 30 training images, instead of the offline augmentation to 1200 images used in prior work (e.g., Chen et al., 2019; Rymarczyk et al., 2020; Ukai et al., 2023). All models were trained with batch size 96 until convergence. The experiments were performed on a workstation with a single NVidia RTX 3090 GPU (24GB VRAM) per model.

6.1 Quantitative Results

Table 1 reports the top-1 accuracy for the tested models on the CUB-200-2011 and Stanford Cars dataset with the Segformer backbone. The Test Accuracy per Epoch curves are visualized in Fig. 5. The Segformer baseline overfit to the small training datasets (17.7% and 1.9% test accuracy), while the ConvNeXt baseline performed better (74.2% and 57.5%). CBM achieved high results quickly, scoring 75.7% on Birds and 79.9% on Cars. ProtoPNet required the most training time, converging only after about 490 epochs, scoring 68% on Birds and 86.4% on Cars. Switching ProtoPNet's prototypes from L_2 to HyperPg improved both training speed and accuracy, converging in 200 epochs with 70.5% and 87.4% test accuracy.

(a) ProtoPNet + HyperPg (b) HyperPgNet - L_{RRC} (c) HyperPgNet + L_{RRC}

Figure 6: Prototype Gradient Maps.

HyperPgNet further improved on this by learning human-defined concepts. HyperPgNet with concept aligned prototypes, but without L_{RRC}, outperformed the other models. It achieved 76.5% accuracy on Birds and 88.6% on Cars after only 40 training epochs. With the "Right for the Right Concept" loss, the model retained the high learning speed and performance, although the test accuracy dropped slightly to 74.1% and 81.2%. This performance loss is offset by increased transparency due to more precise concept prototypes. For additional ablation studies, including number of prototypes, see Appendix B.

6.2 Qualitative Results

HyperPg prototypes do not only enhance the models learning speed and task performance, but also the interpretabilty. Prior work built activations by overlaying the up-sampled latent space activation over the image. Inspired by GradCAM, Fig. 6 shows the gradient of the prototype activations back-propagated to the input image. Class-based prototypes focus on regions relevant for the prediction, but do not follow human defined concepts. With concept-alignment, the prototype focuses more towards the white head of the bird. But as the concept alignment is only encouraged by L_{Clst} and L_{Sep} the prototype still activates on other bright areas on the birds body. With the addition of L_{RRC} to the training regime, the prototypes are forced to focus mainly on the annotated region. Thus, the head prototype ignores the wing area, which is learned by a different prototype. The overall learned concept prototypes fit more precisely to the images' annotated areas. For additional examples and latent space analyses, please refer to Appendix C.

7 Conclusion

This work introduces HyperPg, a new prototype representation learning a probability distribution on the surface of a hypersphere in latent space. HyperPg adapt to the variance of clusters in latent space and improve training time and accuracy compared to other prototype formulations. HyperPgNet leverages HyperPg to learn human defined concept prototypes, instead of black-box optimized prototypes. The combination of probabilistic prototypes on the hypersphere and concept aligned prototypes allows HyperPgNet to outperform other prototype learning approaches. One limitation HyperPgNet faces are slightly higher computational requirements due to the inclusion of concept annotations and L_{RRC} during training. However, this is offset by an acceleration of the training process regarding epochs and number of samples. Coupled with the increased transparency and interpretability of the model, this makes HyperPgNet a strong contender for scenarios with few training samples and high safety requirements, like medical applications or human-robot interaction.

References

Chaofan Chen, Chaofan Chen, Oscar Li, Oscar Li, Daniel Tao, Chaofan Tao, Daniel Tao, Adam S. Barnett, Alina Barnett, Cynthia Rudin, Cynthia Rudin, Jonathan K. Su, and Jonathan Su. This looks like that: Deep learning for interpretable image recognition. *Neural Information Processing Systems*, 2019. doi: null.

Jonathan Crabbé and Mihaela van der Schaar. Concept activation regions: A generalized framework for concept-based explanations. *Advances in Neural Information Processing Systems*, 35:2590–2607, 2022.

Jonathan Donnelly, A. Barnett, and Chaofan Chen. Deformable protopnet: An interpretable image classifier using deformable prototypes. *Computer Vision and Pattern Recognition*, 2021. doi: 10.1109/cvpr52688.2022.01002.

Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2711–2721, 2023.

Thomas Hillen, Kevin J. Painter, Amanda C. Swan, and Albert D. Murtha. Moments of von mises and fisher distributions and applications. *Mathematical Biosciences and Engineering*, 14(3):673–694, 2017. ISSN 1551-0018. doi: 10.3934/mbe.2017038.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pp. 5338–5348. PMLR, 2020.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: a neural network that explains its predictions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, New Orleans, Louisiana, USA, 2018. AAAI Press. ISBN 978-1-57735-800-8.

Chen Liang, Wenguan Wang, Jiaxu Miao, and Yi Yang. Gmmseg: Gaussian mixture based generative semantic segmentation models. *Advances in Neural Information Processing Systems*, 35: 31360–31375, 2022.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

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 (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861.

Pascal Mettes, Elise Van der Pol, and Cees Snoek. Hyperspherical prototype networks. *Advances in neural information processing systems*, 32, 2019.

Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

Nazanin Moradinasab, Laura S Shankman, Rebecca A Deaton, Gary K Owens, and Donald E Brown. Protogmm: Multi-prototype gaussian-mixture-based domain adaptation model for semantic segmentation. *arXiv preprint arXiv:2406.19225*, 2024.

Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14933–14943, June 2021.

Meike Nauta, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. Pip-net: Patch-based intuitive prototypes for interpretable image classification. *Computer Vision and Pattern Recognition*, 2023. doi: 10.1109/cvpr52729.2023.00269.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016. URL https://arxiv.org/abs/1602.04938.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2662–2670, 2017.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. Protopshare: Prototype sharing for interpretable image classification and similarity discovery. *arXiv preprint arXiv:2011.14340*, 2020.

Dawid Rymarczyk, Łukasz Struski, Michał Górszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieliński. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision*, pp. 351–368. Springer, 2022.

Mikołaj Sacha, Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. Protoseg: Interpretable semantic segmentation with prototypical parts. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1481–1492, 2023.

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*, pp. 618–626, 2017.

Y. Ukai, Tsubasa Hirakawa, Takayoshi Yamashita, and H. Fujiyoshi. This looks like it rather than that: Protoknn for similarity-based classifiers. *International Conference on Learning Representations*, 2023. doi: null.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Bor-Shiun Wang, Chien-Yi Wang, and Wei-Chen Chiu. Mcpnet: An interpretable classifier via multi-level concept prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10885–10894, 2024.

Chong Wang, Yuanhong Chen, Fengbei Liu, Davis James McCarthy, Helen Frazer, and Gustavo Carneiro. Mixture of gaussian-distributed prototypes with generative modelling for interpretable image classification. *arXiv preprint arXiv:2312.00092*, 2023.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

Tianfei Zhou, Yi Yang, Ender Konukoğlu, and Luc Van Goo. Rethinking semantic segmentation: A prototype view. *Computer Vision and Pattern Recognition*, 2022. doi: 10.1109/cvpr52688.2022.00261.

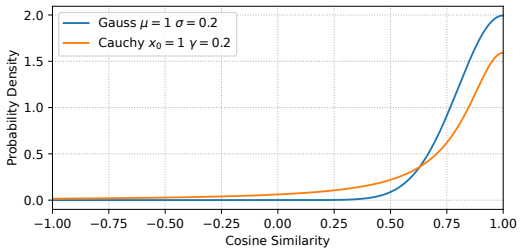


Figure 7: PDF for the Gaussian and Cauchy distribution of cosine similarity values. The Cauchy distribution has heavier tails, avoiding vanishing gradients issues.

A Adapting HyperPg to other Probability Distributions

Subsection 3.3 defines HyperPg prototypes $\boldsymbol{p}^H = (\boldsymbol{\alpha}, \mu, \sigma)$ as a Gaussian Distribution with mean μ and std σ of cosine similarities around an anchor vector α. This idea of learning a distribution of cosine similarity values around an anchor $\boldsymbol{\alpha}$ can be adapted to other distributions. This sections introduces some potential candidates. As early experiments on the CUB-200-2011 dataset showed no significant difference in performance, these sections are relegated to the appendix.

A.1 Cauchy Distribution

One theoretical disadvantage of the Gaussian distribution is the fast approach to zero, which is why a distribution with heavier tails such as the Cauchy distribution might be desirable. The Cauchy distribution's PDF is defined as

$$\mathcal{C}(x; x_0, \gamma) = \frac{1}{\pi\gamma\left(1 + \left(\frac{x - x_0}{\gamma}\right)^2\right)}, \tag{11}$$

with median x_0 and average absolute deviation γ. The HyperPg prototypes with Cauchy are defined as accordingly as $\boldsymbol{p}^{\text{Cauchy}} = (\boldsymbol{\alpha}, x_0, \gamma)$.

Fig. 7 illustrates the PDF of the Gaussian and Cauchy distributions with $\mu = x_0 = 1$ and $\sigma = \gamma = 0.2$, i.e., the main probability mass is aligned with the anchor $\boldsymbol{\alpha}$. The Gaussian distributions PDF quicly approaches zero and stays near constant. This could potentially cause vanishing gradient issues during training. The heavier tails of the Cauchy distribution ensure that for virtually the entire value range of the cosine similarity, gradients could be propagated back through the model. However, experiments on CUB-200-2011 showed no significant performance difference between using HyperPg with the Gaussian or Cauchy distribution.

A.2 Truncated Distributions

The cosine similarity is defined only on the interval $[-1, 1]$. This makes it attractive to also use truncated probability distributions, which are also only defined on this interval. The truncation imposes a limit on the range of the PDF, thereby limiting the influence of large values for the distribution's σ or γ parameter, respectively. The truncated Gaussian pdf $\mathcal{T}_{\text{Gauss}}$ requires the cumulative probability function $\mathcal{G}$ and error function f_{err}, and is defined as

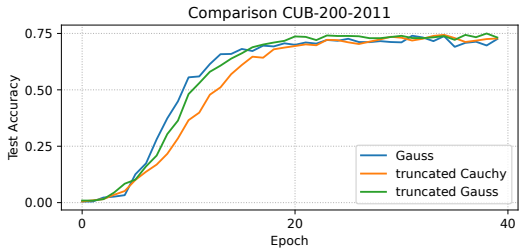


Figure 8: CUB-200-2011 Test Accuracy of HyperPgNet with different probability distributions. The difference in performance is only marginal.

$$f_{\text{err}}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-z^2\right) dz, \tag{12}$$

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{13}$$

$$\mathcal{G}(x, \mu, \sigma) = \frac{1}{2}\left(1 + f_{\text{err}}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right), \tag{14}$$

$$\mathcal{T}_{\text{Gauss}}(x, \mu, \sigma, a, b) = \frac{\mathcal{N}\left(s_{\cos}(\boldsymbol{z}|\boldsymbol{\alpha}); \mu, \sigma\right)}{\mathcal{G}(1, \mu, \sigma) - \mathcal{G}(-1, \mu, \sigma).}, \tag{15}$$

with lower bound a and upper bound b, e.g., for the cosine similarity $a = -1$ and $b = 1$. Similarly, the truncated Cauchy distribution can be applied, which is defined as

$$\mathcal{T}_{\text{Cauchy}}(x, x_0, \gamma, a, b) = \frac{1}{\gamma}\left(1 + \left(\frac{x-x_0}{\sigma}\right)^2\right)^{-1}\left(\arctan\left(\frac{b-x_0}{\gamma}\right) - \arctan\left(\frac{a-x_0}{\gamma}\right)\right)^{-1}. \tag{16}$$

Fig. 8 shows the test accuracy of three HyperPgNet models with Gaussian, truncated Gaussian and truncated Cauchy distribution on the CUB-200-2011 dataset. While difference in test performance and learning speed were minimal on the CUB-200-2011 dataset, further exploration is necessary, as other experiments showed that the concept-alignment on CUB-200-2011 dominates the learning process, lessening the influence of the prototypes.

A.3 von Mises-Fisher Distribution

The von Mises-Fisher distribution (vMF) is the analogue of the Gaussian distribution on the surface of a hypersphere Hillen et al. (2017). The density function f_d of the vMF distribution for a D-dimensional unit-length vector $\boldsymbol{v}$ is defined as

$$f_d(\boldsymbol{v}|\boldsymbol{\alpha}, \kappa) = C_d(\kappa) \exp\left(\kappa \boldsymbol{\alpha}^\top \boldsymbol{v}\right), \tag{17}$$

with mean vector $\boldsymbol{\alpha}$, scalar concentration parameter κ and normalization constant $C_d(\kappa)$. The normalization constant $C_d(\kappa)$ is a complex function and difficult to compute for higher dimensions, which is why, for example, Tensorflow[1] only supports the vMF distribution for $D \leq 5$. However, the vMF distribution is a viable similarity measure when using the unnormalized density function with $C_d(\kappa) = 1$. Working with unnormalized densities highlights the relationship between the normal distribution and the vMF distribution.

Let $\hat{G}$ be the unnormalized PDF of a multivariate Gaussian with normalized mean $\boldsymbol{\alpha}$ and isotropic covariance $\boldsymbol{\sigma}^2 = \kappa^{-1}\boldsymbol{I}$, then it is proportional to the vMF distribution for normalized vectors $\boldsymbol{v}$ with $|\boldsymbol{v}| = 1$, as shown by

[1]Tensorflow API Documentation - Accessed 2024-09-20

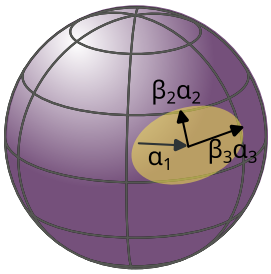


Figure 9: Illustration of the Fisher-Bingham Distribution in $D = 3$

$$\hat{G}(\boldsymbol{v}|\boldsymbol{\alpha}, \kappa) = \exp\left(-\kappa \frac{(\boldsymbol{v}-\boldsymbol{\alpha})^\top (\boldsymbol{v}-\boldsymbol{\alpha})}{2}\right) \tag{18}$$

$$= \exp\left(-\kappa \frac{\boldsymbol{v}^\top \boldsymbol{v} + \boldsymbol{\alpha}^\top \boldsymbol{\alpha} - 2\boldsymbol{v}^\top \boldsymbol{\alpha}}{2}\right) \tag{19}$$

$$= \exp\left(-\kappa \frac{1 + 1 - 2\boldsymbol{v}^\top \boldsymbol{\alpha}}{2}\right) \tag{20}$$

$$= \exp\left(-\kappa \frac{2 - 2\boldsymbol{v}^\top \boldsymbol{\alpha}}{2}\right) \tag{21}$$

$$= \exp\left(-\kappa \frac{1 - \boldsymbol{v}^\top \boldsymbol{\alpha}}{1}\right) \tag{22}$$

$$= \exp\left(\kappa(\boldsymbol{v}^\top \boldsymbol{\alpha} - 1)\right) \tag{23}$$

$$= \exp(\kappa \boldsymbol{v}^\top \boldsymbol{\alpha} - \kappa) \tag{24}$$

$$= \exp(\kappa)^{-1} \exp(\kappa \boldsymbol{v}^\top \boldsymbol{\alpha}) \tag{25}$$

$$\sim \exp\left(\kappa \boldsymbol{v}^\top \boldsymbol{\alpha}\right). \tag{26}$$

Eq. 23 also shows the relationship to the HyperPg similarity with an untruncated Gaussian distribution and prototype mean activation $\mu = 1$.

A.4 Fisher-Bingham Distribution

As the vMF distribution is the equivalent of an isotropic Gaussian distribution on the surface of a hypersphere, the Fisher-Bingham (FB) distribution is the equivalent of a Gaussian with full covariance matrix. Similar to the vMF, the normalization constant is difficult to compute for higher dimensions, but the unnormalized density function remains feasible.

For a D dimensional space, the FB distribution is by a $D \times D$ matrix $\boldsymbol{A}$ of orthogonal vectors $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_D)$, concentration parameter κ and ellipticity factors $[\beta]_{2:D}$ where $\sum_{j=2}^{D} \beta_j = 1$ and $0 \leq 2|\beta_j| < \kappa$. The FB unnormalzied PDF is defined as

$$b(\boldsymbol{v}|\boldsymbol{A}, \kappa, \beta) = \exp\left(\kappa \boldsymbol{\alpha}_1^\top \boldsymbol{v} + \sum_{j=2}^{D} \beta_j \left(\boldsymbol{\alpha}_j^\top \boldsymbol{v}\right)^2\right). \tag{27}$$

The FB distribution's main advantage is the elliptic form of the distribution on the surface of the hypersphere, offering higher adaptability than the other formulations (see Fig. 9. However, the parameter count and constraints are higher.

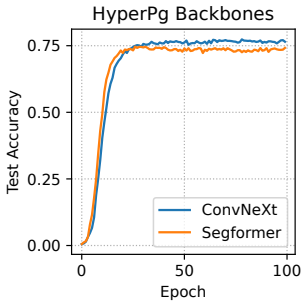


Figure 10: CUB-200-2011 Test Accuracies with different Backbones over 3 random seeds each. The maximum difference in performance is 2%.

A.5 Mixture Models

HyperPg's probabilistic nature lends itself to a mixture formulation. Let the definition of a HyperPg Mixture Prototype be $\boldsymbol{p}^M = (\boldsymbol{\alpha}, \mu, \sigma, \pi)$ with additionally learned mixture weight π. Further, let's define the probability of a latent vector $\boldsymbol{z}$ belonging to a Gaussian HyperPg prototype $\boldsymbol{p}$ as

$$\phi(\boldsymbol{z}|\boldsymbol{p}) = s_{\text{HyperPg}}(\boldsymbol{z}|\boldsymbol{p}). \tag{28}$$

Then the probability of $\boldsymbol{z}$ belonging to class c can be expressed through the mixture of all prototypes $\boldsymbol{p}_c \in \boldsymbol{P}_c$ of class c, i.e.,

$$\phi(\boldsymbol{z}|c) = \sum_{\boldsymbol{p}_c \in \boldsymbol{P}_c} \pi(\boldsymbol{p}_c)\phi(\boldsymbol{z}|\boldsymbol{p}_c). \tag{29}$$

First experiments with mixture of HyperPg prototypes did not show any improvement over the standard formulation. However, this might change with other datasets.

B Ablations

B.1 Backbone Performance

HyperPgNets performance is not dependent on a CNN or Transformer based backbone. Fig. 10 shows the average test accuracy curves for HyperPgNet with concept aligned prototypes on CUB-200-2011 with 3 seeded runs. The maximum differnece in test accuracy between both architectures is 2%.

B.2 Number of prototypes

The number of prototypes per class or concept is an important hyper-parameter for prototype learning models like HyperPgNet. The choice of the number of prototypes controls, how many clusters the model should fit in latent space. For example, ProtoPNet requires 10 prototypes per class or 2000 prototypes in total for the CUB-200-2011 dataset (Chen et al., 2019). ProtoPool is able to merge similar prototypical parts, requiring only 202 prototypes for its best performance (Rymarczyk et al., 2022).

The change from class specific to concept aligned prototypes improves overall test performance, and the added information during training leads to fewer training epochs. Concept Bottleneck Models (CBM) are able to outperform the other models with only 30 learned concepts, or 1 prototype per concept. This indicates that the concept annotations on CUB-200-2011 do not produce multi-modal distributions in latent space, which would require more prototypes per concept. This behavior is confirmed in ablation studies on the number of prototypes for HyperPgNet. Fig. 11 presents the test accuracies for HyperPgNet on full CUB-200-2011 images after 60 training epochs with different numbers of prototypes per concept.

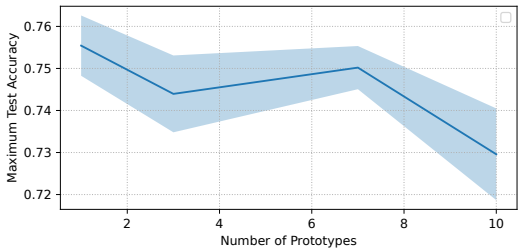


Figure 11: Ablation on number of prototypes in HyperPgNet on the CUB-200-2011 dataset. CUB-200-2011's concept annotations have sufficient granularity for modeling with one prototype per concept. This is in line with CBM's perfomance, which also learns only one prototype per concept.

Figure 12: Concept Prototypes from HyperPgNet

C Extended Interpretability Analysis

C.1 Right for the Right Concept

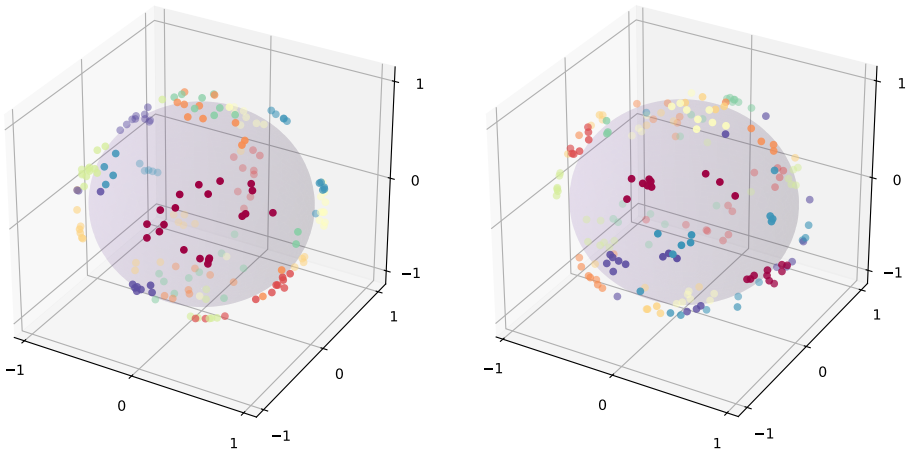
Fig. 12 shows more example prototypes for HyperPgNet. Without L_{RRC}, the prototype do not always correspond to the defined prototype. For the "Orange Body" concept, the prototype also activates on confounding factors in the background. With L_{RRC} during training, the model learns to ignore these factors.

With the car dataset, the effect is even more pronounced. First analyses on the Stanford Cars experiments indicate, that most models focus on the front bumper below the headlights. This part of the car seems to be highly predictive for the class label. Because of this, the HyperPgNet, even with context annotations but without L_{RRC} will focus more on these areas of the image. For example the "bonnet" prototype does not focus on the car bonnet at all. Only when trained with L_{RRC}, the prototype actually focuses on the human defined concept.

C.2 Latent Space Structure

Prototype learning is based on learning structures in the latent space. HyperPg specifically learns Gaussian distributions on the surface of a hypersphere in a high-dimensional latent space. Dimensionality reduction techniques like UMAP (McInnes et al., 2018) aim to preserve global and local structures from a high dimensional space when projecting into a low dimensional one. UMAP supports multiple distance metrices, enabling the projection onto the surface of a 3D sphere.

Fig. 13 illustrates UMAP projections of HyperPg concept-aligned prototypes trained on Stanford Cars. This visualization indicates that HyperPgNet is able to disentangle the different concepts in latent space, as the projection shows no overlap of the different clusters. When trained with L_{RRC}, the prototypes are packed closer together. This could indicate, that the chosen hyper parameter of 20



(a) HyperPgNet - L_{RRC} (b) HyperPgNet + L_{RRC}

Figure 13: UMAP projection of HyperPg concept prototypes learned on Stanford Cars. Each dot represents one prototype, colored by the concept it represents.

prototypes per concept is higher than required as not all concept embeddings have the same diversity in the latent space.

Fig. 14 illustrates one UMAP projection of HyperPg concept-aligend prototypes trained on CUB-200-2011. In comparison to Stanford Cars, the latent space appears less structured. This could explain the higher difficulty associated with this dataset.

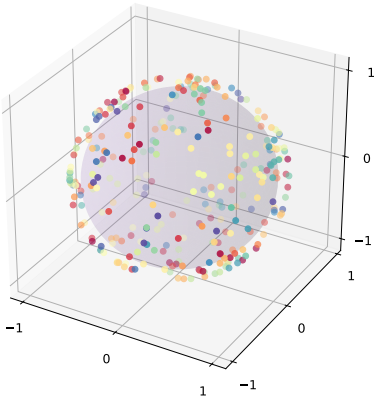


Figure 14: UMAP Projection of HyperPg concept prototypes learned on CUB-200-2011.