This Probably Looks *Exactly* Like That: An Invertible Prototypical Network

Zachariah Carmichael*, Timothy Redgrave*, Daniel Gonzalez Cedre*, and Walter J. Scheirer

University of Notre Dame, Notre Dame IN 46556, USA
`zcarmich@alumni.nd.edu,{tredgrav,dgonza26,wscheire}@nd.edu`
`https://github.com/craymichael/ProtoFlow`
*equal contribution

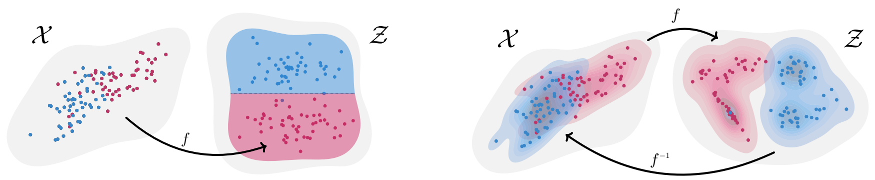
Abstract. We combine concept-based neural networks with generative, flow-based classifiers into a novel, intrinsically explainable, exactly invertible approach to supervised learning. Prototypical neural networks, a type of concept-based neural network, represent an exciting way forward in realizing human-comprehensible machine learning without concept annotations, but a human-machine semantic gap continues to haunt current approaches. We find that reliance on indirect interpretation functions for prototypical explanations imposes a severe limit on prototypes' informative power. From this, we posit that invertibly learning prototypes as *distributions* over the latent space provides more robust, expressive, and interpretable modeling. We propose one such model, called ProtoFlow, by composing a normalizing flow with Gaussian mixture models. ProtoFlow (1) sets a new state-of-the-art in joint generative and predictive modeling and (2) achieves predictive performance comparable to existing prototypical neural networks while enabling richer interpretation.

Keywords: Normalizing flow · Prototypical neural networks · XAI

1 Introduction

Concept-based neural networks offer an attractive way of parsing decisions made by complex systems. By providing explanations in terms of higher-level abstractions, these models provide semantic clarity to both experts and non-experts. While there are myriad different interpretations of the word *concept* in the literature [59], we are interested particularly in prototypical concepts. Prototypes aim to distill traits not directly scrutinizable from the raw data. For instance, the prediction of an image as a bird could be explained by its similarity to a beak prototype [9]. Explainability-by-design guarantees faithful explanations by requiring prototype involvement in decision-making [58]. Such explanations naturally result from the symbolic form of the model [62] rather than *post hoc* correlational analyses [42], a preferable situation for most users [10, 11, 37].

Existing prototypical networks have demonstrated a human-machine semantic similarity gap and often learn irrelevant prototypes [32, 37, 67]. These net-



(a) Other prototypical networks learn points as prototypes and rely on prototype projection.

(b) Our proposed model learns an invertible embedding and distributions as prototypes.

Fig. 1: Existing prototypical networks rely on prototypical points and limiting means of visualizing prototypes. Our proposed approach, ProtoFlow, enables the learning of prototypical distributions with inverses, enabling their exact and efficient visualization—*ProtoFlow inherently enables richer prototype interpretation.*

works typically learn prototypes as *points* in latent space, with limited interpretive power in data space [8, 37, 58, 65]. We hypothesize that this focus on *points* rather than *distributions* is an underlying issue; given a prototype of a blue jay, could we say the model is representing its blue color, its distinctive shape, the texture of its feathers, or something entirely different? It's impossible to tell from just one point. We propose to instead learn prototypical *distributions* over latent space with normalizing flows, capable of generating data while providing exact likelihoods [53, 64]. They are also *fully invertible,* furnishing the latent space with a faithful interpretation back to the data. We leverage this inverse transformation to provide meaningful insight into exactly how the model is learning to represent its training data, in turn allowing an understanding of latent prototypical distributions through their samples' corresponding data-space interpretations. This also removes constraints on learned prototypes [9, 58] and limitations on prototype visualizations [44, 46]. See Fig. 1 for an overview.

In this paper, we propose a novel approach to supervised learning that is interpretable by design by bridging together key ideas from the explainable AI and generative AI literature: generative classifiers and concept-based neural networks. We make the following contributions:

- We develop an approach to learning interpretable latent prototypical distributions with a joint generative and predictive model.
- We propose a diversity loss to reduce prototypical distribution overlap.
- We evaluate our method on various image classification datasets and show state-of-the-art performance at joint predictive and generative modeling.
- We qualitatively and quantitatively analyze explanations generated by the model. We obtain predictive performance comparable to existing prototypical neural networks while enabling richer interpretation.

2 Background

We cover related work on concept-based neural networks and normalizing flows.

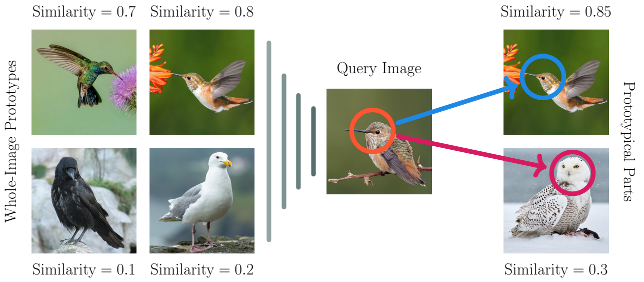


Fig. 2: Examples of whole-image (left) and prototypical parts (right) explanations.

2.1 Concept-Based and Prototypical Neural Networks

Concept-aware models [59] are designed to produce high-level explanations that are more directly meaningful to human examiners. The semantic quality of these explanations is naturally borne by the model's analytic form [58, 62] rather than uninformed *post hoc* correlations [42], which may not be meaningful [61]. *Fully supervised* approaches to concept learning rely on labeled training data that is fully annotated, indicating for example the presence of bone spurs in medical images or the colors of individual body parts in photographs of birds. Notable examples include the concept bottleneck model [40] and the concept embedding model [16]. Such approaches are hindered by the cost and constraints of human labor, restricting their use in *unsupervised* settings. While *weakly supervised* tasks, which don't have annotations, may retain class labels, *fully unsupervised* tasks have neither. This paper will focus on weakly supervised learning with prototypical neural networks [9, 44], the predominant approach in this setting. Though they are related to several other tasks in the literature, including data summarization [27, 58], example-based post hoc explanation [19, 58], and few-shot learning [68], our goal here is to learn prototypes for *intrinsically explainable* case-based reasoning [9, 44].

Prototypical Neural Networks Prototype networks provide local explanations by relating decisions made on input data to discriminative abstractions called *prototypes*. In Fig. 2, we illustrate how images can be explained with (a) whole-image prototypes and (b) prototypical parts corresponding to specific portions of an image. Decisions are made by a composite mapping $\mathcal{X} \xrightarrow{f} \mathcal{Z} \xrightarrow{g} \mathcal{C}$ that sorts the data $\boldsymbol{x} \in \mathcal{X}$ into mutually exclusive classes $c \in \mathcal{C}$. This first involves learning a transformation $\boldsymbol{x} \overset{f}{\mapsto} \boldsymbol{z}$ of our Euclidean data space $\mathcal{X}$ into a latent inner-product space $\mathcal{Z}$ whose parameters specify a set of prototypes related to the class labels. A classifier $\boldsymbol{z} \overset{g}{\mapsto} c$ then makes decisions based on a similarity kernel $\kappa : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ imposed on the learned representations.

Critically, the latent representations $\boldsymbol{z}$ don't exist in the same space as our data points $\boldsymbol{x}$. So, if $\mathcal{X}$ is a space of images for example, nothing so far gives us a way to visually understand how images nor prototypes are represented in $\mathcal{Z}$. This requires an *interpretation* transform $\mathcal{Z} \xrightarrow{h} \mathcal{X}$ going in the other direc-

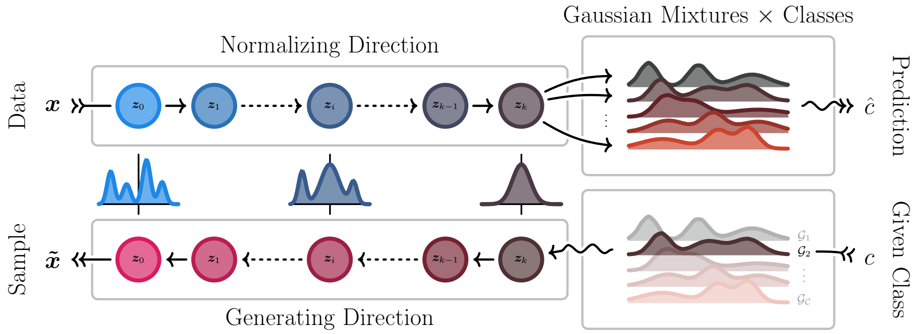


Fig. 3: An overview of the proposed ProtoFlow model that composes a normalizing flow and Gaussian mixture models. In the *normalizing* $\mathcal{X} \xrightarrow{f} \mathcal{Z}$ direction, the invertible composition $f = f_k \circ \cdots \circ f_1$ *pulls back* the structured latent density $p_{\mathcal{Z}}$ to the complex data density $p_{\mathcal{X}}$. The *generating* $\mathcal{X} \xleftarrow{f^{-1}} \mathcal{Z}$ direction *generates* points $\tilde{\boldsymbol{x}} \sim p_{\mathcal{X}}$ implicitly by *pushing forward* samples $\tilde{\boldsymbol{z}} \sim p_{\mathcal{Z}}$ from the latent distribution along the inverse f^{-1}.

tion, and several proposals have been offered in the literature. Li et al. proposed an autoencoder framework with f as the encoder and h as the decoder [44]. This, however, is only capable of approximate reconstructions, with visualization fidelity susceptible to shifts in the latent prototypical distributions. As an alternative, Chen et al. proposed *prototype projection* [9]. Under this scheme—illustrated in Fig. 1a—learned prototype representations are replaced by the image embeddings of their most similar training images. Although this guarantees an association between certain points in $\mathcal{Z}$ and training data from $\mathcal{X}$, the fact that f is typically not injective often results in a human-machine semantic similarity gap [37] when multiple disparate data points are associated with the same latent point. Other approaches choose instead to compute the nearest-neighbor image embeddings to each prototypical point [46, 73]. Point-based approaches such as these have relatively low representation power due to (1) the necessary sparsity of high-dimensional spaces and (2) a dearth of variability measures [74].

Instead of prototypical points, we propose learning prototypes as *distributions* over the latent space as shown in Fig. 1b. Moreover, we recover an intrinsic association between $\mathcal{Z}$ and $\mathcal{X}$ by learning these distributions *invertibly* with *normalizing flows* (*cf.* Sec. 2.2), providing us an implicit mapping $f^{-1} : \mathcal{Z} \to \mathcal{X}$ that interprets and visualizes $\mathcal{Z}$ through the eyes of $\mathcal{X}$.

Related Prototypical Neural Networks Somewhat similarly, Ma et al. [46] propose learning prototypical balls—latent hyperspheres in $\mathcal{Z}$—rather than prototypical points. To interpret a given prototype, they visualize every training image whose latent representation lies inside the ball for that prototype. ProtoSegNet [24] models prototypes as the components of Gaussian mixture models (GMMs) learned over the latent space by a prototypical segmentation network; however, their prototype interpretations are limited to indirect visualizations involving nearest neighbors in the data. MGProto [74] again models prototypes as

GMMs, this time choosing to visualize prototypes by applying prototype projection to the components' mean points. Peters [57] attempts to extract prototypical points from the latent space of a generative model using a normalizing flow generator; these points unfortunately look like noise, so nearest neighbors are used instead. Unlike prior work, we are able to faithfully visualize the full learned prototypical distributions due to our invertible normalizing flow backbone.

Prototypical Parts While full-image prototypical neural networks are intrinsically explainable, the part-level explanations of existing prototypical part neural networks are typically considered post hoc [8, 22, 30, 65, 75]. The exceptions are models like PixPNet [8], ProtINN [57], and ProtoBBNet [24], which either impose constraints on the sizes of their receptive fields or process images by patches (using super-pixels or pre-defined grids, for example). See these papers [9, 22, 75] for more details on post hoc part-level explainability in prototype networks.

2.2 Normalizing Flows

A normalizing flow is an unsupervised density estimator $f : \mathcal{X} \to \mathcal{Z}$ that *invertibly* transforms between the data and latent spaces. Empirical information from data *flows* to the latent space through f, and density inferences are recovered by the inverse $f^{-1} : \mathcal{Z} \to \mathcal{X}$. Given a data distribution $p_{\mathcal{X}}$ and a latent distribution $p_{\mathcal{Z}}$ over $\mathcal{X}$ and $\mathcal{Z}$ respectively, the impact of changing variables [53] on a random variable $\mathbf{x} \sim p_{\mathcal{X}}$ via $f(\mathbf{x}) = \mathbf{z}$ is given by the following formula.

$$p_{\mathcal{X}}(\mathbf{x}) = p_{\mathcal{Z}}(f(\mathbf{x})) \cdot \left| \det \left(\frac{\partial f}{\partial \mathbf{x}} \right) \right| \tag{1}$$

This allows us to formally specify the unknown data distribution $p_{\mathcal{X}}$ as the *pushforward* of a latent distribution $p_{\mathcal{Z}}$, with the inverse transformation f^{-1} responsible for *"pushing"* the latent density $p_{\mathcal{Z}}$, which is typically taken to be well-behaved, *"forward"* onto the more unwieldy data distribution $p_{\mathcal{X}}$. With full knowledge and control over $p_{\mathcal{Z}}$, we can leverage f^{-1} as a generative model for sampling from $p_{\mathcal{X}}$ implicitly through the transformation $f^{-1}(\mathbf{z}) \sim p_{\mathcal{X}}$ where $\mathbf{z} \sim p_{\mathcal{Z}}$. We call this model a "normalizing flow" because f *"normalizes"* the complicated data distribution by *"flowing"* information into the latent space [53].

The mapping f is typically implemented as a composition of invertible functions learned by neural networks. A neural network can be made invertible by constructing a discrete- or continuous-time flow, such as coupling flows, autoregressive flows, linear flows, and planar flows [39]. The model can then be trained by maximum likelihood estimation as in Eq. (1).

2.3 Joint Generative and Predictive Modeling

Normalizing Flows Invertible neural networks have already seen use for classification tasks [5, 17, 36, 48] partly due to their memory efficiency during training. However, our focus here is on joint generative and predictive modeling. Fetaya et al. [18] composed a normalizing flow (Glow [38]) with a latent GMM to perform joint generation and prediction. Atanov et al. [4] and Izmailov et al. [34] take similar approaches but with alternative normalizing flows [14, 25] and extensions to semi-supervised learning. Ardizzone et al. [3, 47] continue this trend by adding an information bottleneck term to their loss that balances between performance and robustness. Rather than training a separate classifier (as with *hybrid* models), these approaches classify by estimating the latent-conditional class likelihood $p_{\mathcal{Z}}(y \mid \boldsymbol{z})$. Although our approach happens to overlap with these models, we are motivated directly by intrinsically explainable modeling using *prototypes.* Notably, we choose to learn *multiple* prototypical distributions in the latent space as the base for a normalizing flow, allowing for potentially greater mode coverage [80].

Other Generative Approaches Both hybrid and joint generative/predictive models have been based on other generating methods than normalizing flows. Some notable hybrid models have used deep belief networks [31,50], autoencoders [29], generative adversarial models (GANs) [35], and diffusion models [7, 77]. Closely related, some supervised tasks involve generating output that doesn't match the input data distribution. Notable examples here—based on GANs [35] and diffusion models [7,77]—have been applied to object detection [15], image classification [79], forecasting [76], and next-frame video prediction [78]. Joint generative/predictive modeling dates back to 1996 when Revow et al. proposed generative models built from B-splines for recognizing and generating handwritten digits [63]. More recent approaches have involved variational autoencoders [66], diffusion models [28,43], and scored-based generative classifiers [81]. Unfortunately, none of these generators provide an exact likelihood, an important consideration for explainability. They also fail to enforce a faithful association between the latent and data distributions—only providing approximate inverse mappings from latent to data space—unlike the exact inverse $f^{-1} : \mathcal{Z} \to \mathcal{X}$ granted by normalizing flows (see Fig. 1). We therefore only consider normalizing flow generators.

3 Invertible Prototypical Networks

We propose an intrinsically interpretable approach to learning that bridges together generative classification with concept-based networks—two key ideas from the explainable AI and generative AI literature. As described in Sec. 2, we aim to invertibly learn latent prototypical distributions. In this section, we first describe the normalizing-flow backbone of our architecture. Then, we detail how prototypical distributions are learned over the latent space. Finally, we discuss our training methodology for ProtoFlow, our proposed architecture for joint predictive and generative modeling. An overview of ProtoFlow is given in Fig. 3.

Normalizing Flow Backbone We base our normalizing flow on DenseFlow [26], a state-of-the-art unconditional density estimator [54]. DenseFlow is comprised of invertible Glow-like [38] modules using cross-unit coupling and densely connected blocks fused with Nyström self-attention, increasing expressiveness by incrementally augmenting latent vectors with noise. We replace DenseFlow's *unconditional* latent distribution with *conditional* distributions as follows.

Prototypical Gaussian Mixture Classifier Whereas other prototypical neural networks usually learn latent *points* to represent their prototypes, our prototypes are given by *probability distributions* learned over the latent space. By leveraging the inverse mapping $f^{-1} : \mathcal{Z} \to \mathcal{X}$, these can be reinterpreted as probability distributions over data, providing a direct, faithful, and accurate visualizations of learned prototypes. For each class $c \in \mathcal{C} = \{1, \ldots, C\}$, we specify a K-component GMM $\mathcal{G}_c$ whose components represent prototypical distributions, so that each class has K associated prototypical distributions. The components are weighted by $\boldsymbol{\pi}_c = (\pi_{c,1}, \ldots, \pi_{c,K})$.

$$\mathcal{G}_c = \sum_{k=1}^{K} \pi_{c,k} \mathcal{G}_{c,k} = \sum_{k=1}^{K} \pi_{c,k} \mathcal{N}(\boldsymbol{\mu}_{c,k}, \boldsymbol{\Sigma}_{c,k}) \tag{2}$$

The class-conditional likelihood is given below, where σ is the softmax function.

$$p_{\mathcal{Z}}(\boldsymbol{z} \mid y) = \sum_{k=1}^{K} \sigma(\pi_{y,k}) \mathcal{N}(\boldsymbol{z}\,;\, \boldsymbol{\mu}_{y,k}, \boldsymbol{\Sigma}_{y,k}) \tag{3}$$

As is often done [23], we constrain $\mathcal{G}_c$ by (1) asserting $\boldsymbol{\Sigma}_{c,k}$ are diagonal, (2) clipping $\boldsymbol{\Sigma}_{c,k}$ above zero, and (3) enforcing $\boldsymbol{\pi}_c^{\top} \boldsymbol{\pi}_c = 1$. Applying Bayes' theorem to Eq. (3), we derive the expression below for the data-conditional class likelihood.

$$p_{\mathcal{X}}(y \mid \boldsymbol{x}) = \frac{\sum_{k=1}^{K} \sigma(\pi_{y,k}) \mathcal{N}(f(\boldsymbol{x})\,;\, \boldsymbol{\mu}_{y,k}, \boldsymbol{\Sigma}_{y,k})}{\sum_{c=1}^{C} \sum_{k=1}^{K} \sigma(\pi_{c,k}) \mathcal{N}(f(\boldsymbol{x})\,;\, \boldsymbol{\mu}_{c,k}, \boldsymbol{\Sigma}_{c,k})} \tag{4}$$

Relating this back to our discussion in Sec. 2, we can induce the similarity kernel κ at a given point $\boldsymbol{z} \in \mathcal{Z}$ by looking at the mean $\boldsymbol{\mu}_{c,k}$ of the prototype that maximizes the class-conditional likelihood $\kappa(\boldsymbol{z}, \boldsymbol{\mu}_{c,k}) = p_{\mathcal{Z}}(\boldsymbol{z} \mid y = c; \pi_{c,k}, \boldsymbol{\mu}_{c,k}, \boldsymbol{\Sigma}_{c,k})$. The classifier g can be written $g(f(\boldsymbol{z})) = \operatorname{argmax}_{y \in \mathcal{C}} p_{\mathcal{X}}(y \mid f(\boldsymbol{z}))$ using Eq. (4). Finally, and most interestingly, our interpretation function h is simply the inverse transform $h = f^{-1}$ induced by the normalizing flow. This imposes no additional constraints on the learned prototypes, unlike prior work [9, 44, 46, 73].

Training We train our model to maximize the categorical cross entropy, denoted $\mathcal{L}_{\text{CE}}$, with auxiliary loss terms. Since it is known to improve model robustness, we adapt the proposed *consistency regularization* loss from [34]. This encourages the model to be *invariant* to certain perturbations or augmentations of the training data by penalizing the model for predicting two different classes $\dot{y} \neq \ddot{y}$ for two perturbations $\dot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{x}}$ of the same data point $\boldsymbol{x} \in \mathcal{X}$, expressed below.

$$\mathcal{L}_{\text{CR}}(\dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}) = -\log p_{\mathcal{X}}(\dot{\boldsymbol{x}} \mid \ddot{y}) = -\log p_{\mathcal{Z}}\big(f(\dot{\boldsymbol{x}}) \mid y = \ddot{y}\big) - \log \left| \det \left(\frac{\partial f}{\partial \dot{\boldsymbol{x}}} \right) \right| \tag{5}$$

To help support diversity and reduce information overlap between prototypes in each class, we penalize the components within each mixture based on their squared Hellinger distance $\mathcal{H}^2$. However, high-dimensional spaces' asymptotic sparsity—sometimes referred to as the *curse of dimensionality*—limits the usefulness of $\mathcal{H}^2$: as the expected discernibility between points decreases, gradients tend to vanish. To mitigate this, we propose a *modified* divergence $\widetilde{\mathcal{H}}^2$ rescaled based on the embedding dimension $d = \dim(\mathcal{Z})$. Given two multivariate Gaussians $\mathcal{N}_1 = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}_2 = \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, the modified divergence $\widetilde{\mathcal{H}}^2(\mathcal{N}_1, \mathcal{N}_2)$ between them—which is not a metric—is given below [45].

$$1 - \frac{\det(\boldsymbol{\Sigma}_1 \boldsymbol{\Sigma}_2)^{1/4d}}{\det\left(\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}\right)^{1/2d}} \exp\left(-\frac{1}{8d}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\mathsf{T} \left(\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}\right)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right) \tag{6}$$

This is bounded within $[0, 1] \subseteq \mathbb{R}$. The symmetry of Eq. (6) can be leveraged to avoid repetitive computations, yielding the following form for the diversity loss.

$$\mathcal{L}_{\text{DIV}}(\mathcal{G}) = \frac{-2}{CK(K-1)} \sum_{c=1}^{C} \sum_{i=1}^{K-1} \sum_{j=i}^{K} \widetilde{\mathcal{H}}^2(\mathcal{G}_{c,i}, \mathcal{G}_{c,j}) \tag{7}$$

With hyperparameters λ_{CR} and λ_{DIV}, the final training objective is given below.

$$\mathcal{L} = \mathcal{L}_{\text{CE}}\big(p_{\mathcal{X}}(y \mid \boldsymbol{x}), y\big) + \lambda_{\text{CR}} \mathcal{L}_{\text{CR}}(\dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}) + \lambda_{\text{DIV}} \mathcal{L}_{\text{DIV}}(\mathcal{G}) \tag{8}$$

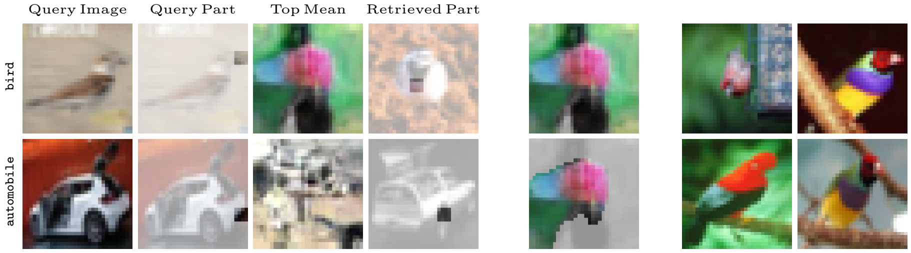
The collection of mixtures $\mathcal{G}$ can be trained either by stochastic gradient descent (SGD) or expectation maximization (EM). Empirically, however, we find that SGD results in better classification performance than EM. We explore K-means as an initialization strategy of the means of Gaussians. We initialize the K mean parameters of the mixture $\mathcal{G}_c$ responsible for modeling $p_{\mathcal{Z}}(\boldsymbol{z} \mid y = c)$ with the K-means clusters of the embeddings $\{f(\boldsymbol{x}) \mid (\boldsymbol{x}, c) \in (\mathcal{X}, \mathcal{C})\}$. During training, we keep an exponential moving average (EMA) of the model parameters to reduce training time. This is a variant of Polyak averaging [60].

Pruning Not all prototypes may end up being useful: they may be redundant, noisy, semantically meaningless, or contribute negligibly to prediction performance. Pruning such prototypes not only reduces not only the number of parameters in the model but also the quantity of information a user must digest to parse an explanation. We propose pruning prototypes based on the weights $\pi_{c,k}$ learned on their respective mixture components. Prototypes with weights below a minimal threshold ε—selected using Otsu's method [52]—are discarded.

Prototypical Parts We propose a new approach to explaining prototypical parts.

The original prototypical part network ProtoPNet [9] enforces a dimensional correspondence between $\mathcal{X}$ and $\mathcal{Z}$. They find prototypical parts by first computing an element-wise similarity between each embedded image $\boldsymbol{z}$ of shape $d \times \ell' \times w'$ and a given prototypical point $\boldsymbol{p}_k$ that lives in a d-dimensional affine subspace of $\mathcal{Z}$, resulting in a similarity map for each prototype. These maps get upsampled to the image dimension $\ell \times w$ using bicubic interpolation. The top 5% scoring pixels in the original image are then finally selected as prototypical parts.

We take a different approach. Let $\mathbf{x} \sim p_{\mathcal{X}}$ be a random variable distributed according to our data-generating density, and let $\boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots \boldsymbol{x}_N\} \subseteq \mathcal{X}$ be a



(a) Queries, prototype means, and most likely parts. **(b)** Mean point. **(c)** Similar data points.

Fig. 4: (a)*"This looks like that"*-style explanations of `bird` (top row) and `automobile` (bottom row) image classification decisions. Rather than using training samples as prototypes, ProtoFlow learns prototype distributions directly over the latent space, leading to the *"bird/car-adjacent"* images in the third column. The fourth column shows the most-likely dataset image part for each prototypical distribution. (b) The mean point image of the `bird` prototype with a bird-like figure segmented from the background. (c) Human-picked images from CIFAR-10 that qualitatively match this prototype image.

query image. We introduce the notation ${}_{i}^{j}\boldsymbol{x}_{i+\ell}^{j+w}$ to mean the block of pixels from the image $\boldsymbol{x}$ with coordinates ranging $\{i, \dots i+\ell\}$ horizontally and $\{j, \dots j+w\}$ vertically. We then compute a heatmap $\mathcal{M}_{c,k}$ for each prototype $\mathcal{G}_{c,k}$ as follows.

$$\bar{\boldsymbol{x}} := \mathbb{E}[\mathbf{x}] = \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \boldsymbol{x}_i \tag{9}$$

$${}_{i}^{j}\bar{\boldsymbol{x}}_{i+\ell}^{j+w} \leftarrow {}_{i}^{j}\boldsymbol{x}_{i+\ell}^{j+w} \tag{10}$$

$${}_{i}^{j}(\mathcal{M}_{c,k})_{i+\ell}^{j+w} \leftarrow p_{\mathcal{Z}}\left(f\left(\bar{\boldsymbol{x}}\right) \mid \mathcal{G}_{c,k}\right) \tag{11}$$

On line (9), we begin with an image $\bar{\boldsymbol{x}}$ computed as the pixel-wise average of every image $\boldsymbol{x}_i$ in the training data. On line (10), we replace the image patch in $\bar{\boldsymbol{x}}$ specified by the rectangle with lower-left corner at (i, j) and upper-right corner at $(i+\ell, j+w)$ with the corresponding patch of pixels from a given query image $\boldsymbol{x}$, leaving the rest of $\bar{\boldsymbol{x}}$ as it was. This results in an image whose background is averaged across the entire dataset containing a single patch from a query image. On line (11), we then evaluate $p_{\mathcal{Z}}(f(\bar{\boldsymbol{x}}) \mid \mathcal{G}_{c,k})$ and replace the corresponding patch in $\mathcal{M}_{c,k}$ with the prototype-conditional likelihood of this patched image. Like ProtoPNet, our network's receptive field is the full input, so it would be inappropriate to attribute a subset of pixels to a particular prototype. However, we can emphasize the top 5% of pixels to indicate the most influential image parts on a classification decision. Since we assume independence between image patches, this is an approximate, post hoc method in the same vein as ProtoPNet.

4 Experiments and Analysis

We evaluate ProtoFlow on a variety of image classification datasets used to evaluate prior joint predictive and generative modeling approaches: MNIST [13],

Dataset	Res	Model	Proto-Based	Flow-Based	Acc ↑	BPD ↓	ECE ↓	MCE ↓
MNIST	28×28	ProtoFlow (Ours)	✓	✓	99.36	**0.535**	0.006	0.587
		FlowGMM [34]	✗	✓	**99.63**	—	**0.004***	—
		Fetaya et al. [18]	✗	✓	99.30	1.00	—	—
		SCNF-GLOW [4]	✗	✓	88.44	1.15	—	—
		SCNF-GMM [4]	✗	✓	83.10	1.14	—	—
CIFAR-10	32×32	ProtoFlow (Ours)	✓	✓	**91.54**	3.95	0.083	**0.494**
		IB-INN ($\gamma \to \infty$) [3]	✗	✓	91.28	17.3	0.81	13.9
		IB-INN ($\gamma = 1$) [3]	✗	✓	89.73	5.25	0.54	3.25
		FlowGMM [34]	✗	✓	88.44	—	**0.038***	—
		Fetaya et al. [18]	✗	✓	84.00	**3.53**	—	—
		KMEx [21]	✓	✗	85.3	—	—	—
		ProtoPNet [9, 21]	✓	✗	84.9	—	—	—
		FLINT [55]	✓	✗	84.7	—	—	—
		ProtoVAE [20]	✓	✗	84.6	—	—	—
CIFAR-100	32×32	ProtoFlow (Ours)	✓	✓	**69.80**	5.03	**0.292**	**0.637**
		IB-INN ($\gamma \to \infty$) [3]	✗	✓	66.22	18.4	0.62	16.8
		IB-INN ($\gamma = 1$) [3]	✗	✓	57.43	**4.93**	0.58	7.04
Flowers-102	64×64	ProtoFlow (Ours)	✓	✓	59.80	13.5	0.141	0.295
Oxford-IIIT Pet	64×64	ProtoFlow (Ours)	✓	✓	53.58	4.89	0.459	0.736

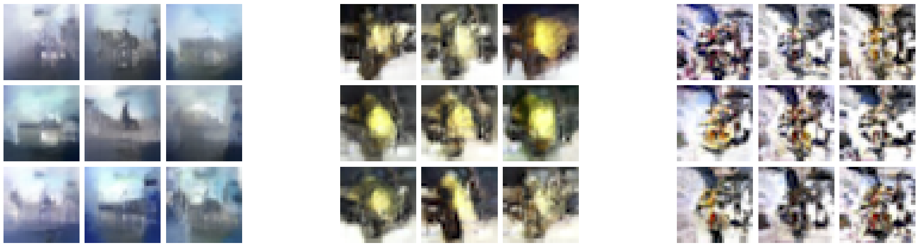
Table 1: Results of normalizing-flow-based joint generative and predictive models across image classification tasks. Results reported from other prototypical neural networks are shown for CIFAR-10. ProtoFlow achieves state-of-the-art accuracy while retaining highly competitive density estimation and calibration scores. *The result is reported with temperature-based scaling of GMM variances [34].

CIFAR-10 [41], and CIFAR-100 [41]. We also consider more challenging datasets, including Flowers-102 [51], Oxford-IIIT Pet [56], and CUB-200-2011 [72].

Setup We perform transfer learning by initializing the DenseFlow backbone with weights pre-trained for unconditional density estimation on 64×64 and 32×32 ImageNet [12]. Images are appropriately resized to these resolutions. As DenseFlow is stochastic, we employ Monte Carlo sampling to reduce its variance. We also use test-time augmentation. All hyperparameters and other reproducibility details are provided in the supplemental material. Code, configurations, and trained models are available at `https://github.com/craymichael/ProtoFlow`.

Predictive and Generative Performance We evaluate model performance using metrics that assess both predictive performance and generative performance. Classification accuracy is used to quantify the former. For the latter, we consider bits per dimension (BPD), which is a common density estimation metric for normalizing flows. BPD describes how many bits would be needed to encode a particular image in the modeled distribution [71]. The results across all datasets are given in Tab. 1. For existing generative classifiers based on normalizing flows, our approach achieves state-of-the-art accuracy and competitive BPD on all tasks. We also establish baselines on additional challenging datasets. ProtoFlow outperforms other prototypical neural networks that report results on CIFAR-10.

Uncertainty and Calibration Reliable uncertainty quantification is critical in many applications, especially when algorithmic modeling leads to high-stakes decision-making. In turn, we evaluate the reliability of predictive uncertainty



(a) `ship` (b) `frog` (c) `cat`

Fig. 5: Mean points (center) and generated samples (periphery) from prototype distributions learned on CIFAR-10 with consistency loss and a truncation value of 1.

by using the expected (ECE) and maximum (MCE) calibration These measures quantify the expected and maximum deviation of the predicted probabilities from the ground truth distribution, respectively. As shown in Tab. 1, our approach has relatively low calibration error across all datasets.

Interpretability As discussed in Sec. 3, learned prototype distributions can be visualized faithfully and accurately by applying the inverse transform f^{-1} to points sampled from the latent space. We can influence the diversity and quality of these samples by drawing from truncated versions of the learned Gaussian distributions. This technique—sometimes called the *"truncation trick"* [6]—is often used when sampling from GANs. Mean points and samples from prototypical distributions taken using this process are shown in Fig. 5. For a more thorough analysis on the effects of truncation, please see the supplementary materials.

On CIFAR-10, we explore our proposed prototypical parts approach. See Fig. 4 for examples of prototypical part explanations—the beak of a bird and the bumper of a car with high and low likelihood, respectively, are emphasized by ProtoFlow. The relevance ordering test is a quantitative measure of how well a heatmap attributes individual pixels according to prototype likelihoods [22]. Starting from a completely random image, pixels are added back to the random image one at a time in descending order according to the heatmap $\mathcal{H}^k$. As each pixel is added back, the likelihood of $\boldsymbol{z}$ conditioned on prototypical distribution k is evaluated. This procedure is averaged over each class-specific prototype over the data. The faster the likelihood increases, the better the heatmap is. Fig. 6

Dataset	$\mathcal{S}_{\text{ROB}}$ ↑	$\mathcal{S}_{\text{DIV}}$ ↑
MNIST	0.995	0.606
CIFAR-10	0.879	0.503
CIFAR-100	0.950	0.666

Table 2: ProtoFlow robustness and diversity scores for learned prototypical distributions.

Dataset	% Pruned	Acc ↑	BPD ↓	ECE ↓	MCE ↓
MNIST	0	99.39	0.535	0.006	0.587
	54.0	96.26	1.13	0.025	0.591
CIFAR-10	0	91.54	3.95	0.083	0.494
	78.0	91.13	3.95	0.087	0.614
CIFAR-100	0	69.80	5.03	0.292	0.637
	89.9	69.63	5.02	0.294	0.637

Table 3: ProtoFlow pruning results.

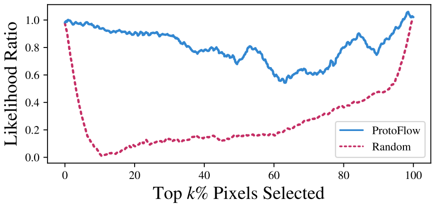


Fig. 6: Relevance ordering test for ProtoFlow trained on CIFAR-10. Our heatmap approach discovers important image parts with respect to each prototypical distribution as substantiated by the likelihood gap with random heatmaps.

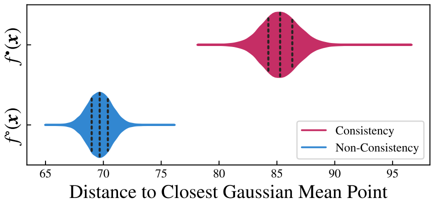


Fig. 7: Distances from each embedded point to its nearest prototypical mean point plotted for a model trained *with* (in red) and *without* (in blue) consistency regularization. Quartiles are shown as dashed lines.

shows the results of this test—our approach identifies important image parts substantially faster than random. Interestingly, a uniformly random image has high likelihood, which lowers quickly as the pixels are added back randomly. For a deeper understanding, please see the supplemental visualizations.

Robustness to Noise Are prototype distributions learned by ProtoFlow robust to noise? If so, then predicted prototypes should not be affected by small variations in the data. Let $\mathbf{e} \sim \mathcal{N}(0, s^2\boldsymbol{I})$ be multivariate Gaussian noise and $\mathbf{x} \sim p_{\mathcal{X}}$ be random data. The maximum likelihood prototype for $\mathbf{x}$ is given by $ML(\mathbf{x})$ below.

$$ML(\mathbf{x}) = \underset{(c,k)\in\mathcal{C}\times\mathcal{K}}{\operatorname{argmax}}\ p_{\mathcal{Z}}\left(f(\mathbf{x}) \mid \mathcal{G}_{c,k}\right). \tag{12}$$

We set the noise variance $s = 0.2$ for experiments, inspired by prior work [8,33]. We define a robustness score, conceptually similar to a stability score [33], below.

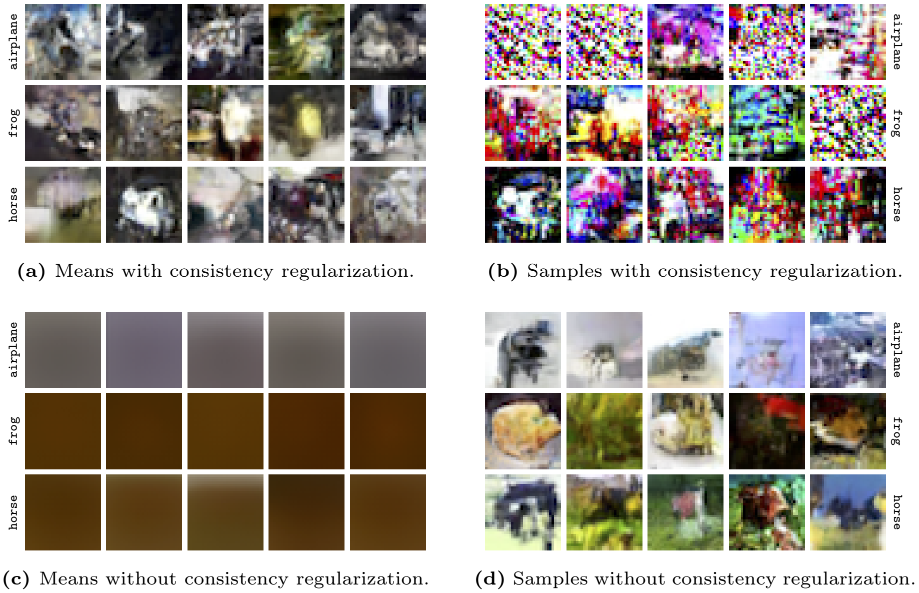
$$\mathcal{S}_{\text{ROB}} = \mathbb{E}_{\mathcal{G}}\left[\!\left[ML(\mathbf{x}) = ML(\mathbf{x}+\mathbf{e})\right]\!\right] \tag{13}$$

The Iverson brackets $[\![\cdot]\!]$ above evaluate to 1 or 0 when the statement within the brackets is true or false, respectively. S_{ROB} represents the expected proportion of the data whose most likely prototype is stable under perturbation. Tab. 2 shows ProtoFlow is fairly robust, with $\mathcal{S}_{\text{ROB}} \gtrapprox 90\%$ across datasets.

Diversity We propose a new score to quantify the diversity of the learned prototypical distributions based on the Shannon entropy H taken over prototypes.

$$\mathcal{S}_{\text{DIV}} = \frac{H\left(ML(\mathbf{x}); \mathcal{G}\right)}{\log(KC)} = \frac{-\mathbb{E}_{\mathcal{G}}\left[p\left(ML\left(\mathbf{x}\right)\right)\log p\left(ML(\mathbf{x})\right)\right]}{\log(KC)} \tag{14}$$

A lower score indicates redundant or uninformative prototypical distributions, and higher scores suggest diversity among prototypes. In the worst case, there is a single prototypical distribution that maximizes the likelihood of every training sample. In the best case, samples are evenly spread out across prototype distributions, yielding $H(ML(\mathbf{x}); \mathcal{G}) = \log(KC)$. We normalize by this maximum



(a) Means with consistency regularization.

(b) Samples with consistency regularization.

(c) Means without consistency regularization.

(d) Samples without consistency regularization.

Fig. 8: A visual comparison of prototypical distributions on CIFAR-10 learned with and without consistency regularization. The consistency mean points **(a)** seem more interpretable than the uninformative means **(c)** learned without consistency. Despite this, samples with consistency **(b)** are poor compared to non-consistency samples **(d)**.

entropy to bound $\mathcal{S}_{\text{DIV}}$ between 0 and 1. Tab. 2 shows $\mathcal{S}_{\text{DIV}}$ varies between datasets and is far from ideal, suggesting ProtoFlow is learning redundant prototypes.

Prototype Pruning We evaluate the impact of prototype pruning on predictive and generative modeling performance. We hypothesized that models might learn uninformative prototypes that could be removed without substantial performance degradation. We confirm this hypothesis in Tab. 3. In particular, $\sim$90% of prototypes can be removed from a model trained on CIFAR-100 with <0.2% drop in accuracy. This reinforces our prior result regarding prototype diversity.

Impact of Consistency Loss To test the impact of the consistency regularization term $\mathcal{L}_{\text{CR}}$, we compare a version of ProtoFlow trained with consistency regularization $f^{\bullet}$ to a version f° trained without it. Both models are trained on CIFAR-10 with the same hyperparameters. We visualize mean points and samples from a random subset of the corresponding prototypes and classes. The visualizations of $f^{\bullet}$ and f° are shown in Fig. 6 and Fig. 7, respectively. Immediately, we see an interesting pattern. The prototypes learned *with* $\mathcal{L}_{\text{CR}}$ have good-looking mean point visualizations, but naively sampling from those distributions produces noise—this motivated our use of truncated sampling in models trained with $\mathcal{L}_{\text{CR}}$ (*cf.* Fig. 5). Contrast this with the uninformative and blurry prototype means learned *without* this loss term, which surprisingly yield high-quality samples that (1) correspond visually with their respective class and (2) vary predictably and sensibly between classes. Since this behavior is consistent across prototypes within each model, we hypothesized that $\mathcal{L}_{\text{CR}}$ affected the spatial distribution of embedded latent points relative to the learned prototypes. For each data point $\boldsymbol{x}_i \in \mathcal{X}$, we compute $\min_{\mathcal{G}_{c,k}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{c,k}\|$ and plot these distributions in Fig. 7. The model f° trained *without* $\mathcal{L}_{\text{CR}}$ tended to embed the data points significantly closer in $\mathcal{Z}$ to the means of prototypical distributions than $f^{\bullet}$ did. This helps explain why samples from f° appear more like real images.

This further highlights the flaw with point-based prototype explanations. The noise in the samples visualized from $f^{\bullet}$ suggests the regions of $\mathcal{Z}$ corresponding to those prototypes are not actually informative about the model—at least to a human. Comparing these samples to their mean points reveals just how misleading a single point can be, possibly even signaling instances of overfitting. Conversely, despite uninformative mean points, the samples taken from f° are consistently and undeniably more meaningful.

Additional Analyses and Results Please see the supplemental material for analyses of GMM initialization, results on CUB-200-2011, ablation of auxiliary loss terms, effect of Monte Carlo sampling, prototype visualizations, and more.

5 Discussion

Our proposed prototypical neural network ProtoFlow achieves state-of-the-art performance on joint generative predictive modeling tasks. By learning prototypes as latent-space probability distributions using normalizing flows, it simultaneously delivers rich, faithful visualizations and precise uncertainty quantification. We also establish baselines on more complex datasets than prior work[1].

A limitation of our implementation is the 64 × 64 pixel maximum on input image resolution, an artifact of the pre-trained DenseFlow models available. We also fail to fully exploit invertibility during backpropagation—intermediate activations can be recomputed during the backward pass instead of being stored during the forward pass [5, 17, 36, 48]. Since ProtoFlow is vulnerable to training data biases, we recommend appropriate safety and fairness precautions [1, 2, 49].

Future work should find it straight-forward to extend ProtoFlow to semi-supervised settings by optimizing the joint likelihood over labeled and unlabeled data. The number of prototypical distributions should be selected in a more principled way—*e.g.*, using information theory [69, 70]. Even by applying techniques to avoid overfitting, such as dropout, data augmentation, and weight regularization, it proved difficult to avoid a generalization gap on the Oxford-IIIT Pets and Flowers-102 datasets. Subsequent research should consider more aggressive data augmentation and increasing the resolution of images used as input to ProtoFlow. We are hopeful that this new avenue for prototype learning in neural networks will lead to further innovations that open the black box and close the human-machine semantic similarity gap.

[1] IB-INN [47] was evaluated on ImageNet, but these results may not be reproducible. See the supplemental material for more detail and further discussion.

Acknowledgements

Timothy Redgrave's travel was supported by the U.S. Department of Defense (Contract No. W52P1J-20-9-3009). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Department of Defense or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation here on.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D.: Concrete problems in ai safety. arXiv preprint arXiv:1606.06565 (2016)
3. Ardizzone, L., Mackowiak, R., Rother, C., Köthe, U.: Training normalizing flows with the information bottleneck for competitive generative classification. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 7828–7840. Curran Associates, Inc. (2020), `https://proceedings.neurips.cc/paper_files/paper/2020/file/593906af0d138e69f49d251d3e7cbed0-Paper.pdf`
4. Atanov, A., Volokhova, A., Ashukha, A., Sosnovik, I., Vetrov, D.: Semi-conditional normalizing flows for semi-supervised learning. In: 1st workshop on Invertible Neural Networks and Normalizing Flows (ICML 2019) (2019)
5. Behrmann, J., Grathwohl, W., Chen, R.T., Duvenaud, D., Jacobsen, J.H.: Invertible residual networks. In: International conference on machine learning. pp. 573–582. PMLR (2019)
6. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), `https://openreview.net/forum?id=B1xsqj09Fm`
7. Cao, H., Tan, C., Gao, Z., Xu, Y., Chen, G., Heng, P.A., Li, S.Z.: A survey on generative diffusion models. IEEE Transactions on Knowledge and Data Engineering (2024)
8. Carmichael, Z., Lohit, S., Cherian, A., Jones, M.J., Scheirer, W.J.: Pixel-grounded prototypical part networks. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 4768–4779 (January 2024)
9. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.: This looks like that: Deep learning for interpretable image recognition. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Neural Information Processing Systems, NeurIPS. pp. 8928–8939 (2019), `https://proceedings.neurips.cc/paper/2019/hash/adf7ee2dcf142b0e11888e72b43fcb75-Abstract.html`
10. Davoodi, O., Mohammadizadehsamakosh, S., Komeili, M.: On the interpretability of part-prototype based classifiers: a human centric analysis. Scientific Reports **13**(1), 23088 (2023)

11. Dawoud, K., Samek, W., Eisert, P., Lapuschkin, S., Bosse, S.: Human-centered evaluation of xai methods. In: 2023 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 912–921. IEEE (2023)
12. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA. pp. 248–255. IEEE Computer Society (2009). https://doi.org/10.1109/CVPR.2009.5206848
13. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
14. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real NVP. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=HkpbnH9lx
15. Ehsani, K., Mottaghi, R., Farhadi, A.: Segan: Segmenting and generating the invisible. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6144–6153 (2018)
16. Espinosa Zarlenga, M., Barbiero, P., Ciravegna, G., Marra, G., Giannini, F., Diligenti, M., Shams, Z., Precioso, F., Melacci, S., Weller, A., et al.: Concept embedding models: Beyond the accuracy-explainability trade-off. Advances in Neural Information Processing Systems **35**, 21400–21413 (2022)
17. Etmann, C., Ke, R., Schönlieb, C.: iunets: Learnable invertible up- and downsampling for large-scale inverse problems. In: 30th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2020, Espoo, Finland, September 21-24, 2020. pp. 1–6. IEEE (2020). https://doi.org/10.1109/MLSP49062.2020.9231874
18. Fetaya, E., Jacobsen, J., Grathwohl, W., Zemel, R.S.: Understanding the limitations of conditional generative models. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=r1lPleBFvH
19. Filho, R.M., Lacerda, A.M., Pappa, G.L.: Explainable regression via prototypes. ACM Transactions on Evolutionary Learning **2**(4), 1–26 (2023)
20. Gautam, S., Boubekki, A., Hansen, S., Salahuddin, S., Jenssen, R., Höhne, M., Kampffmeyer, M.: ProtoVAE: A trustworthy self-explainable prototypical variational model. Advances in Neural Information Processing Systems **35**, 17940–17952 (2022)
21. Gautam, S., Boubekki, A., Höhne, M., Kampffmeyer, M.C.: Prototypical self-explainable models without re-training. Transactions on Machine Learning Research (2024), https://openreview.net/forum?id=HU5DOUp6Sa
22. Gautam, S., Höhne, M.M.C., Hansen, S., Jenssen, R., Kampffmeyer, M.: This looks more like that: Enhancing self-explaining models by prototypical relevance propagation. Pattern Recognition **136**, 1–13 (2023). https://doi.org/10.1016/j.patcog.2022.109172
23. Gepperth, A., Pfülb, B.: Gradient-based training of gaussian mixture models for high-dimensional streaming data. Neural Processing Letters **53**(6), 4331–4348 (2021)
24. Gerstenberger, M., Maaß, S., Eisert, P., Bosse, S.: A differentiable gaussian prototype layer for explainable fruit segmentation. In: 2023 IEEE International Conference on Image Processing (ICIP). pp. 2665–2669. IEEE (2023)

25. Grathwohl, W., Chen, R.T.Q., Bettencourt, J., Sutskever, I., Duvenaud, D.: FFJORD: free-form continuous dynamics for scalable reversible generative models. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=rJxgknCcK7
26. Grcić, M., Grubišić, I., Šegvić, S.: Densely connected normalizing flows. Advances in Neural Information Processing Systems **34**, 23968–23982 (2021)
27. Gurumoorthy, K.S., Dhurandhar, A., Cecchi, G., Aggarwal, C.: Efficient data representation by selecting prototypes with importance weights. In: 2019 IEEE International Conference on Data Mining (ICDM). pp. 260–269. IEEE (2019)
28. Han, X., Zheng, H., Zhou, M.: CARD: classification and regression diffusion models. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022 (2022), http://papers.nips.cc/paper_files/paper/2022/hash/72dad95a24fae750f8ab1cb3dab5e58d-Abstract-Conference.html
29. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
30. Hesse, R., Schaub-Meyer, S., Roth, S.: FunnyBirds: A synthetic vision dataset for a part-based analysis of explainable AI methods. In: IEEE/CVF International Conference on Computer Vision, ICCV. pp. 1–18. IEEE (2023)
31. Hinton, G.E.: To recognize shapes, first learn to generate images. Progress in brain research **165**, 535–547 (2007)
32. Hoffmann, A., Fanconi, C., Rade, R., Kohler, J.: This looks like that... does it? Shortcomings of latent space prototype interpretability in deep networks. In: ICML Workshop on Theoretic Foundation, Criticism, and Application Trend of Explainable AI (2022). https://doi.org/10.48550/ARXIV.2105.02968
33. Huang, Q., Xue, M., Huang, W., Zhang, H., Song, J., Jing, Y., Song, M.: Evaluation and improvement of interpretability for self-explainable part-prototype networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2011–2020 (2023)
34. Izmailov, P., Kirichenko, P., Finzi, M., Wilson, A.G.: Semi-supervised learning with normalizing flows. In: International Conference on Machine Learning. pp. 4615–4630. PMLR (2020)
35. Jabbar, A., Li, X., Omar, B.: A survey on generative adversarial networks: Variants, applications, and training. ACM Computing Surveys (CSUR) **54**(8), 1–49 (2021)
36. Jacobsen, J., Smeulders, A.W.M., Oyallon, E.: i-revnet: Deep invertible networks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net (2018), https://openreview.net/forum?id=HJsjkMb0Z
37. Kim, S.S., Meister, N., Ramaswamy, V.V., Fong, R., Russakovsky, O.: Hive: Evaluating the human interpretability of visual explanations. In: European Conference on Computer Vision. pp. 280–298. Springer (2022)
38. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. pp. 10236–10245 (2018), https://proceedings.

neurips.cc/paper/2018/hash/d139db6a236200b21cc7f752979132d0-Abstract.html

39. Kobyzev, I., Prince, S.J., Brubaker, M.A.: Normalizing flows: An introduction and review of current methods. IEEE transactions on pattern analysis and machine intelligence **43**(11), 3964–3979 (2020)
40. Koh, P.W., Nguyen, T., Tang, Y.S., Mussmann, S., Pierson, E., Kim, B., Liang, P.: Concept bottleneck models. In: International conference on machine learning. pp. 5338–5348. PMLR (2020)
41. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
42. Leavitt, M.L., Morcos, A.: Towards falsifiable interpretability research. In: NeurIPS Workshop on ML-Retrospectives, Surveys & Meta-Analyses. pp. 1–15. arXiv (2020). https://doi.org/10.48550/ARXIV.2010.12016
43. Li, A.C., Prabhudesai, M., Duggal, S., Brown, E., Pathak, D.: Your diffusion model is secretly a zero-shot classifier. In: IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023. pp. 2206–2217. IEEE (2023). https://doi.org/10.1109/ICCV51070.2023.00210
44. Li, O., Liu, H., Chen, C., Rudin, C.: Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
45. Llorente, L.P.: Statistical inference based on divergence measures. CRC Press (2006)
46. Ma, C., Zhao, B., Chen, C., Rudin, C.: This looks like those: Illuminating prototypical concepts using multiple visualizations. Advances in Neural Information Processing Systems **36** (2024)
47. Mackowiak, R., Ardizzone, L., Kothe, U., Rother, C.: Generative classifiers as a basis for trustworthy image classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2971–2981 (2021)
48. Mangalam, K., Fan, H., Li, Y., Wu, C.Y., Xiong, B., Feichtenhofer, C., Malik, J.: Reversible vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10830–10840 (2022)
49. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. ACM Comput. Surv. **54**(6) (jul 2021). https://doi.org/10.1145/3457607
50. Ng, A., Jordan, M.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Advances in neural information processing systems **14** (2001)
51. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian conference on computer vision, graphics & image processing. pp. 722–729. IEEE (2008)
52. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics **9**(1), 62–66 (1979). https://doi.org/10.1109/TSMC.1979.4310076
53. Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. The Journal of Machine Learning Research **22**(1), 2617–2680 (2021)
54. Image generation | papers with code, https://paperswithcode.com/task/image-generation, accessed: 2024-02-15
55. Parekh, J., Mozharovskyi, P., d'Alché Buc, F.: A framework to learn with interpretation. Advances in Neural Information Processing Systems **34**, 24273–24285 (2021)

56. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.: Cats and dogs. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3498–3505. IEEE (2012)
57. Peters, M.: Extending explainability of generative classifiers with prototypical parts. Master's thesis, University of Twente (2022)
58. Poché, A., Hervier, L., Bakkay, M.C.: Natural example-based explainability: A survey. In: Longo, L. (ed.) Explainable Artificial Intelligence - First World Conference, xAI 2023, Lisbon, Portugal, July 26-28, 2023, Proceedings, Part II. Communications in Computer and Information Science, vol. 1902, pp. 24–47. Springer (2023). `https://doi.org/10.1007/978-3-031-44067-0_2`
59. Poeta, E., Ciravegna, G., Pastor, E., Cerquitelli, T., Baralis, E.: Concept-based explainable artificial intelligence: A survey. arXiv preprint arXiv:2312.12936 (2023)
60. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM journal on control and optimization **30**(4), 838–855 (1992)
61. Poursabzi-Sangdeh, F., Goldstein, D.G., Hofman, J.M., Wortman Vaughan, J.W., Wallach, H.: Manipulating and measuring model interpretability. In: Proceedings of the 2021 CHI conference on human factors in computing systems. pp. 1–52 (2021)
62. Räz, T.: Ml interpretability: Simple isn't easy. Studies in history and philosophy of science **103**, 159–167 (2024). `https://doi.org/10.1016/j.shpsa.2023.12.007`
63. Revow, M., Williams, C., Hinton, G.: Using generative models for handwritten digit recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **18**(6), 592–606 (1996). `https://doi.org/10.1109/34.506410`
64. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International conference on machine learning. pp. 1530–1538. PMLR (2015)
65. Sacha, M., Jura, B., Rymarczyk, D., Struski, Ł., Tabor, J., Zieliński, B.: Interpretability benchmark for evaluating spatial misalignment of prototypical parts explanations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38 (2024)
66. Schott, L., Rauber, J., Bethge, M., Brendel, W.: Towards the first adversarially robust neural network model on MNIST. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), `https://openreview.net/forum?id=S1EHOsC9tX`
67. Sinhamahapatra, P., Heidemann, L., Monnet, M., Roscher, K.: Towards human-interpretable prototypes for visual assessment of image classification models. In: Radeva, P., Farinella, G.M., Bouatouch, K. (eds.) Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2023. vol. 5, pp. 878–887. SCITEPRESS (2023). `https://doi.org/10.5220/0011894900003417`
68. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Advances in neural information processing systems **30** (2017)
69. Still, S., Bialek, W.: How many clusters? an information-theoretic perspective. Neural computation **16**(12), 2483–2506 (2004)
70. Sugar, C.A., James, G.M.: Finding the number of clusters in a dataset: An information-theoretic approach. Journal of the American Statistical Association **98**(463), 750–763 (2003)
71. Theis, L., van den Oord, A., Bethge, M.: A note on the evaluation of generative models. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), `http://arxiv.org/abs/1511.01844`

72. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
73. Wan, Q., Wang, R., Chen, X.: Interpretable object recognition by semantic prototype analysis. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 800–809 (January 2024)
74. Wang, C., Chen, Y., Liu, F., McCarthy, D.J., Frazer, H., Carneiro, G.: Mixture of gaussian-distributed prototypes with generative modelling for interpretable image classification. arXiv preprint arXiv:2312.00092 (2023)
75. Wolf, T.N., Bongratz, F., Rickmann, A.M., Pölsterl, S., Wachinger, C.: Keep the faith: Faithful explanations in convolutional neural networks for case-based reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38 (2024)
76. Yan, T., Zhang, H., Zhou, T., Zhan, Y., Xia, Y.: Scoregrad: Multivariate probabilistic time series forecasting with continuous energy-based generative models. arXiv preprint arXiv:2106.10121 (2021)
77. Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., Yang, M.H.: Diffusion models: A comprehensive survey of methods and applications. ACM Comput. Surv. **56**(4) (nov 2023). `https://doi.org/10.1145/3626235`
78. Yang, R., Srivastava, P., Mandt, S.: Diffusion probabilistic modeling for video generation. Entropy **25**(10), 1469 (2023)
79. Yang, Y., Fu, H., Aviles-Rivero, A.I., Schönlieb, C.B., Zhu, L.: Diffmic: Dual-guidance diffusion network for medical image classification. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 95–105. Springer (2023)
80. Zhong, P., Mo, Y., Xiao, C., Chen, P., Zheng, C.: Rethinking generative mode coverage: A pointwise guaranteed approach. Advances in Neural Information Processing Systems **32** (2019)
81. Zimmermann, R.S., Schott, L., Song, Y., Dunn, B.A., Klindt, D.A.: Score-based generative classifiers. In: Deep Generative Models and Downstream Applications Workshop (2021)