Journal of Machine Learning Research 24 (2023) 1-49 Submitted 10/21; Revised 4/23; Published 11/23

ProtoShotXAI: Using Prototypical Few-Shot Architecture for Explainable AI

Samuel Hess SHESS@ARIZONA.EDU
Gregory Ditzler GREGORY.DITZLER@GMAIL.COM
EpiSys Science
Poway, CA 92064, USA

Editor: Florence d'Alche-Buc

Abstract

Unexplainable black-box models create scenarios where anomalies cause deleterious responses, thus creating unacceptable risks. These risks have motivated the field of eXplainable Artificial Intelligence (XAI) which improves trust by evaluating local interpretability in black-box neural networks. Unfortunately, the ground truth is unavailable for the model's decision, so evaluation is limited to qualitative assessment. Further, interpretability may lead to inaccurate conclusions about the model or a false sense of trust. We propose to improve XAI from the vantage point of the user's trust by exploring a black-box model's latent feature space. We present an approach, ProtoShotXAI, that uses a Prototypical few-shot network to explore the contrastive manifold between nonlinear features of different classes. A user explores the manifold by perturbing the input features of a query sample and recording the response for a subset of exemplars from any class. Our approach is a locally interpretable XAI model that can be extended to, and demonstrated on, few-shot networks. We compare ProtoShotXAI to the state-of-the-art XAI approaches on MNIST, Omniglot, and ImageNet to demonstrate, both quantitatively and qualitatively, that ProtoShotXAI provides more flexibility for model exploration. Finally, ProtoShotXAI also demonstrates novel explainability and detectability on adversarial samples.

Keywords: Explainable AI, XAI, Local Interpretability, Few-Shot Learning.

1. Introduction

Large-scale neural networks have made promising strides to reduce the classification error over large volumes of data (He et al., 2016; Silver et al., 2016; Van Den Oord et al., 2016). For example, residual networks have surpassed human-level accuracy on ImageNet (He et al., 2016; Russakovsky et al., 2015a). Policy networks have defeated professional humans in the game of Go (Silver et al., 2016), and WaveNet can automatically generate natural human-sounding speech from text (Van Den Oord et al., 2016). Unfortunately, these black-box models often lack human interpretable reasoning for their inferences because of the millions of weights, biases, and nonlinear activation functions in their architectural complexity. One example in recent literature presented evidence that the top ImageNet classification models were biased on textures rather than shape bias, demonstrating that prior explanations and interpretations of these models might be incorrect (Geirhos et al., 2019). Although textures may be sufficient for some tasks, the unexpected exploitation of texture leaves the model

vulnerable to undesirable predictions, especially with potential exposure to real-world data anomalies (Alcorn et al., 2019).

The concern of deleterious predictions from black-box models has triggered research efforts in eXplainable AI (XAI) that can be broadly organized into (1) the development of explainable models that are comparable in prediction performance to their black-box counterparts, and (2) the development of techniques to derive explanations from existing black-box models (Bodria et al., 2023; Adadi and Berrada, 2018). Due to additional constraints of human interpretability, explainable models often struggle to achieve comparable performance with respect to their black-box counterparts. Additionally, explainable models are designed for a specific task, which limits their ability to extend to other tasks or datasets. In contrast, techniques that derive explanations from black-box models are support tools that can provide expectations on prediction behavior. The scope of explanations are often limited under the assumption(s) of the model and are not easily human interpretable. For example, some XAI approaches explain black-box models by weighting the importance of pixels. Unfortunately, these pixel attribution weights are still ambiguous because they require a human interpreter to extrapolate higher-level meanings from the weights (e.g., large pixel weights around a cat's ears can indicate the shape of the ears are important high-level features). As discussed by Guidotti (2021), this extrapolation of meaning only measures the adherence of the weights to the preconceived perception of individuals and leads to one of the core challenges with explaining black-box models: *there is no ground truth for the explanation itself*. In the cat example, large pixel weights around the ears could indicate that the shape of the ear is important for classification; however, the pixel weights could also indicate that the ear's texture is truly important. Unfortunately, both explanations are convenient for a human interpreter, but not necessarily correct. There could be many underlying true reasons for the pixel weights around the cat's ears. For example, it could be the ratio of power levels between pixels, the color, or many other possibilities. The actual rationale for the pixel weight is unknown and a challenge in XAI.

A common approach to explain a model when ground truth does not exist is to involve actual human assessment through a survey. For example, Jeyakumar et al. (2020) performed Mechanical Turk surveys to determine user preference for explanations of deep neural networks (DNN) for image, text, audio, and sensor classification. Their study evaluated various state-of-the-art XAI against their proposed explain-by-example approach (namely, ExMatchina). It showed users preferred ExMatchina for most tasks (e.g., image, audio, and sensor). Although the study provided an interesting insight into human preference, there is a disconnect between a human's explanation preference and the decision of the black-box model. That is, the empirical evaluation of user preferences can exploit the user's confirmation bias (i.e., what the user already believes are important features).

Since there is no ground truth for the explanation of a black-box model's prediction and human assessment of feature importance is inherently biased, we propose in this work to refocus XAI towards tractable tools for model exploration and trust. Further, ensuring trust in the model is even more challenging when data availability is limited. Few-shot learning scenarios are one such example of these limited data tasks, where only a "few" labeled data are available Snell et al. (2017b). In these scenarios, the notion of trusting a model becomes extremely challenging from a data perspective but also challenging from the ability to explain a model since there are limited data to learn. It is against this

background that we have developed a novel algorithm that is more amenable to exploring a model's manifold space w.r.t input feature manipulations. Specifically, we present ProtoShotXAI, a flexible black-box exploration approach that takes advantage of a model's trained feature representation layer, the weights of the classification layer (if available), and a Prototypical few-shot architecture to cross-compare a given query sample to class prototypes. ProtoShotXAI allows a human-in-the-loop to manipulate an image under test and/or the support set to explore the changes to the feature representation. This user involvement provides more flexibility in testing prior hypotheses and a general exploration of model behavior.

In this work, we evaluate our novel approach against many state-of-the-art model explanations, including Grad-CAM, SHAP, LIME, RISE, and ExMatchina. We demonstrate that our approach can produce attribution maps that are popular with the XAI research literature and achieves comparable qualitative performance. Further, we present several novel experiments that allow human-in-the-loop driven model exploration that is unique to our approach. Most notably, deep few-shot architectures require both a query sample and a set of support samples to evaluate w.r.t. the query, and our approach can account for both the query and support set. The primary contributions in this work can be summarized as:

- A novel few-shot architecture, namely ProtoShotXAI, is presented as an XAI approach that merges local interpretability and prototype methods. ProtoShotXAI leverages human-in-the-loop interaction to drive a more useful cross-comparison of feature representations. Our method has a broader impact in testing different hypotheses from a model's decision.

- ProtoShotXAI is model-free because our approach is not specific to a particular neural network. Hence, our approach has broader applicability than some existing works in XAI.

- This work presents one of the first approaches to bridge XAI and few-shot architectures to increase the interpretability of few-shot neural networks and conventional models.

- Development of a tractable approach for model exploration and trust for black-box models, providing more insightful interpretations of real-world datasets.

- Publicly available code & data at `https://github.com/samuelhess/ProtoShotXAI/` which includes cross-comparisons between common XAI approaches with consistent representations as well as four unique experiments for XAI: adversarial MNIST, revolving six, Omniglot XAI, and a multi-class ImageNet example.

The remainder of this manuscript is organized as follows. Section 2 provides a background of recent and related works on XAI and few-shot learning. Section 3 describes the ProtoShotXAI approach, optimization, and use for learning localized regions of importance in images that explain a model's decision from a feature representation. Section 4 presents experiments on three benchmark datasets against several state-of-the-art XAI approaches. Section 5 discusses the findings from our experiments and variants of ProtoShotXAI. Finally, Section 6 highlights the contributions, broader impacts, summary of the results and avenues for future research.

2. Related Work

This section reviews the relevant related work and is organized into two subsections. Section 2.1 summarizes the related XAI approaches with a focus on locally interpretable approaches that are compared against our work in the experiment section. Section 2.2 presents related work on few-shot learning architectures, specifically the evolution of the prototypical few-shot architecture that is leveraged in our approach. There are a few related works that address both XAI and Siamese few-shot networks which are reviewed at the end of Section 2.2.

2.1 eXplainable AI (XAI)

XAI can be broadly partitioned into two main categories: (1) approaches that are inherently explainable, and (2) approaches that derive explanations from black-box models (Bodria et al., 2023; Adadi and Berrada, 2018). In this work, we focus exclusively on explaining existing black-box models that can be applied to DNNs. Within these methods, related work falls into the subcategory of locally interpretable approaches. Popular approaches in this subcategory include saliency maps (Simonyan et al., 2014), Grad-CAM++ (Chattopadhay et al., 2018), LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017), RISE (Petsiuk et al., 2018), and Anchors (Ribeiro et al., 2018).

Saliency maps are visualized as heat maps that indicate feature importance and the maps are used to determine correlated areas in an image that are valuable to a model's prediction. Saliency maps are common amongst all the approaches mentioned above and originally evolved from literature in gradient-based methods. Similar to how a neural network trains, gradient methods take the derivative of an input w.r.t. an output class activation. The derivative provides a direction of feature manipulations that, when applied to the input features, would result in a positive activation gain for being "more like" the respective class. More formally, given a trained black-box classifier that transforms a D-dimensional sample $\mathbf{x} \in \mathbb{R}^D$ into an estimated vector of C classification scores $\tilde{\mathbf{y}} \in \mathbb{R}^C$, the gradient map $\mathbf{z} \in \mathbb{R}^D$ is simply the partial derivative of the loss w.r.t. the input features $\mathbf{z} = \frac{\partial \mathbf{y}_c}{\partial \mathbf{x}}$. Unfortunately, unregularized gradient-based changes to the input often exploit the neural network's function by providing unique, unrealistic, and/or unexpected combinations of noisy gradients rather than providing clear insight into an input feature's value w.r.t the network's prediction. As a result, many variants of gradient-based approaches apply spatial averaging and other regularization techniques to constrain the gradients and reduce their noise. This regularization and smoothing helps to provide consistency to the scale of a feature in a spatial region which, in turn, improves human interpretability (Selvaraju et al., 2017; Chattopadhay et al., 2018).

In contrast to gradient-based methods, Local Interpretable Model-agnostic Explanations (LIME) adds small perturbations to a given sample's input features and creates a linear model between the perturbations and the output predictions (Ribeiro et al., 2016). Specifically, a sample $\mathbf{x}$ is transformed into a dataset $X' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_N\}$ where $\mathbf{x}'_i$ are N perturbations of $\mathbf{x}$. The dataset is passed through the classifier to produce a dataset of input samples and classification score pairings $T = \{(\mathbf{x}'_1, \mathbf{y}'_1), \ldots, (\mathbf{x}'_N, \mathbf{y}'_N)\}$. A linear model can then be defined as $y'_i(c) = \mathbf{z}(c) \cdot \mathbf{x}'_i$ where the linear coefficients $\mathbf{z}$ represent the map of feature importance for each class c and can be learned using multinomial logistic regression.

In contrast to allowing gradients to dictate the direction of changes to the input, LIME effectively tests local input-output regions around a given sample more systematically. LIME assumes that, whereas the total input-output space is often highly complex and nonlinear, the small perturbation region around a given sample under test can likely be approximated as linear. This assumption, along with the significant variance of coefficients $\mathbf{z}$ w.r.t the choice of input perturbations X', have spawned many variants of LIME (Shankaranarayana and Runje, 2019; Zafar and Khan, 2019; Peltola, 2018; ElShawi et al., 2019; Bramhall et al., 2020). Of particular noteworthiness, the original authors of LIME also created an alternative rules-based approach called Anchors. In Anchors, the features of an input sample are again perturbed; however, unlike LIME, they are perturbed intelligently with a multi-armed bandit to produce a set of "anchors" that consistently hold for an output decision regardless of other (non-anchor) feature values. Instead of the feature attribution weights produced by other approaches, Anchors produce a set of logical feature boundaries for a decision.

Randomize Input Sample for Explanation (RISE) is an approach similar to LIME (Petsiuk et al., 2018). RISE perturbs the input data with a binary mask. As in LIME, a sample $\mathbf{x}$ is transformed into a dataset $X' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_N\}$ where $\mathbf{x}'_i$ are N *randomly masked* representations of $\mathbf{x}$ (i.e., $\mathbf{x}'_i = \mathbf{x} \odot \mathbf{m}_i$ where $\odot$ is the Hadamard product and $\mathbf{m}_i$ is the binary mask). The masked dataset is passed through the classifier to produce a dataset of input samples and classification score pairings $T = \{(\mathbf{x}'_1, \mathbf{y}'_1), \ldots, (\mathbf{x}'_N, \mathbf{y}'_N)\}$. Unlike LIME, however, the data are not used to train a regression model. Instead, masks are weighted by the score for the class of interest, and the cumulative results are normalized as defined by

$$\mathbf{z} = \frac{1}{\mathbb{E}\left[\mathbf{m}\right] N} \sum_{i=1}^{N} y'_i(c)\mathbf{m}_i$$

where $\mathbb{E}\left[\mathbf{m}\right]$ is the expected value across the masks and is set externally via a hyperparameter (Petsiuk et al., 2018). Setting a value of 0.5 would indicate that a pixel in the mask is a 0 or 1 with equal probability. It is worth noting that LIME also provides both positive and negative attribution weights, whereas RISE is strictly positive attribution. The negative feature attributions can be interpreted as pixels (or regions) causing an image to deviate from the class of interest. Note that the class of interest need not be the class predicted by the model. We subsequently refer to this as the *evidence* of a counterfactual, where a counterfactual is the necessary alterations required to change the inferred class (Vermeire et al., 2022).

Often contrasted with LIME and RISE is Shapley Additive Explanations (SHAP), which is an approach that focuses on the Shapley values from cooperative game theory. Shapley values define the importance of an input feature as the likelihood that the outcome would be different if a model was trained without that feature. Naturally, training numerous models with different features removed is often intractable, so most SHAP approaches use clever ways to approximate Shapley values with different underlying assumptions on the type of data or model (Sundararajan et al., 2017; Smilkov et al., 2017; Erion et al., 2021). Notably, Grad-SHAP and DeepSHAP (also known respectively as GradientExplainer and DeepExplainer in the SHAP code libraries) are the versions of SHAP specifically designed for DNNs that we evaluate in this work. Grad-SHAP use a subset of N training samples

for background reference $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ and define the explainable map as

$$\mathbf{z} = \mathop{\mathbb{E}}_{\mathbf{x}' \sim T, \alpha \sim U(0,1)} \left[(\mathbf{x} - \mathbf{x}'_i) \frac{\partial \mathbf{f}(\mathbf{x}'_i + \alpha(\mathbf{x} - \mathbf{x}'_i))}{\partial \mathbf{x}} \right]$$

where $\mathbf{x}'_i$ is sampled from a subset of training samples, α is sampled from a uniform distribution, and the partial derivative is the gradient at a feature layer $\mathbf{f}$ of the network w.r.t the input. The function $\mathbf{f}(\mathbf{x}'_i + \alpha(\mathbf{x} - \mathbf{x}'_i))$ represents a random hybrid between the input features under test $\mathbf{x}$ and a sampled background $\mathbf{x}'_i$. $\mathbf{f}$ can be the output of any feature layer up until the classification layer; however, the feature map becomes spatially coarser as the convolutional layers become coarser (and deeper layers in a CNN are often coarser). This observation means that the selection of a layer towards the middle should be preferred over a layer towards the end of the network.

Another related subcategory to our work are prototype methods. XAI, through prototypes, attempts to provide the user with a different but relatable sample (i.e., a "prototype") that is an archetypical representation. Many of these approaches compute a distance between the prototypes and other samples in either the feature space of the input training data or a high-level feature manifold of a trained black-box model (Bien and Tibshirani, 2011; Kim et al., 2016; Koh and Liang, 2017; Papernot and McDaniel, 2018; Gurumoorthy et al., 2019). The distance metric is used to find a representative class sample that is close to the query sample. For example, ExMatchina uses the final feature layer (i.e., the layer next to the classification layer) to compute the cosine similarity between the query sample and training samples to find the N closest matches within that multi-dimensional manifold. Formally, given a trained black-box model and a sample $\mathbf{x}$, the exemplary sample $\mathbf{x'}_e$ that best explains $\mathbf{x}$ is

$$\mathbf{x'}_e = \mathop{\arg\max}_{\mathbf{x}' \in T} \frac{\mathbf{f}^{\mathsf{T}}(\mathbf{x})\mathbf{f}(\mathbf{x}')}{\|\mathbf{f}(\mathbf{x})\|_2^2 \cdot \|\mathbf{f}(\mathbf{x}')\|_2^2}$$

where the $\mathbf{x}'$ is from the the set of N training samples $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ and $\mathbf{f}(\cdot)$ is the K-dimensional feature representation of $\mathbf{x}$. Intuitively, if $\mathbf{x}$ is in the training set ($\mathbf{x} \in T$) then $\mathbf{x}'_e$ is itself (i.e., $\mathbf{x}'_e = \mathbf{x}$).

2.2 Few-Shot Learning

In addition to related work in XAI, ProtoShotXAI leverages network architectures developed in the few-shot learning literature. Few-shot is a subcategory of classification where the goal is to categorize (with arbitrarily high accuracy) a set of unlabeled query samples into specific classes after being presented with a set of labeled support samples that contain one (or a "few") exemplary sample(s) from each class. In contrast to conventional classification, where many training samples are available for each class, few-shot training datasets have a limited number of training samples per class. For example, the MNIST dataset has roughly 6,000 samples per class. In contrast, the comparable few-shot character dataset (known as Omniglot) has only 20 samples per class (Section 4 presents further details of these datasets). Additionally, a benchmark evaluation of few-shot networks is performed on classes that are mutually exclusive from the training classes. Note that this is in contrast to conventional classification tasks that are only evaluated on classes in the training set. A few-shot task

requires that one (or a "few") exemplary samples of classes are provided during testing and conventional classification provides no such supplementary information.

The concept of few-shot has been around for decades in machine learning literature (Fink, 2004; Fei-Fei et al., 2006); however, many recent neural network developments towards few-shot have emerged in response to the creation of benchmark datasets. In response to the aforementioned Omniglot dataset, Siamese networks (Koch et al., 2015) created an architecture of two identical networks (i.e., networks with tied weights) where two different samples are presented at the input of each network. During training, Siamese networks compute a Euclidean distance at the feature layer of the network before the final sigmoid activation. The final activation is zero if two samples are from the same class and one if they are different (similar to the signature verification approach presented in Bromley et al. (1994)). The network was evaluated by randomly subsampling one exemplary sample from 20 random Omniglot classes and classifying the remainder of the samples from the 20 classes (known as 1-shot, 20-way evaluation). It achieved 93.4% classification accuracy.

Subsequent neural network developments extended the capability from one-shot to few-shot, improved the classification accuracy, and expanded the application to additional datasets. Notably, the authors of Matching Networks developed a network training scheme called episodic training where subsets of the training data are repeatedly (and randomly) selected to mimic the few-shot evaluation (Vinyals et al., 2016). Similar to Siamese networks, Matching Networks uses tied weights and computes a distance metric at the feature layer. Each exemplary sample produces a distance w.r.t the sample under test and a neural attention mechanism selects the most likely class. They demonstrated state-of-the-art classification performance on the Omniglot benchmark as well as the more complex mini-ImageNet dataset introduced by Ravi and Larochelle (Ravi and Larochelle, 2017). Most metric-based approaches continue to use the episodic training scheme, including Prototypical Networks. The authors of Prototypical Networks demonstrated that they could achieve even better performance than Matching Networks with a simplified architecture (similar to Siamese Networks), a Euclidean distance metric at the feature layer, and a softmax over the distances (Snell et al., 2017a). Many few-shot architectures have been developed since Prototypical Networks, including metric architectures (Ye and Guo, 2018; Scott et al., 2018), generative architectures (Wang et al., 2018; Antoniou et al., 2018), meta-learning architectures (Yoon et al., 2018; Finn et al., 2018), or some combination of these approaches; however, Chen et al. showed many of the few-shot methods have comparable results when using the same neural network backbone (Chen et al., 2019).

There have been a few related approaches that address both XAI and Siamese few-shot networks. Zhang et al. (2018) investigated Siamese networks for sound searching and maximized the node activations in a fully connected layer of a trained network. By sonifying the difference between the input pairs that maximized node activations in the network, they provided some human interpretability to hear the sound features that provide the best match. This approach is similar to the gradient based XAI methods applied to sound features and Siamese networks. Two other approaches (Utkin et al., 2020; Ye et al., 2020), explain Siamese networks by training a separate model that are inherently explainable. For example, Utkin et al. (2020) uses a post-hoc autoencoder that learns to weight the embeddings of the Siamese network to maximize similarity, reconstructing the inputs with the modified weights provides interpretable insight into the important features.

In contrast, Chen et al. (2021) and Fedele et al. (2022) use post-hoc methods that perturb the input of the Siamese network without having to train a separate explainable network. Specifically, Fedele et al. (2022) use automated segmentation to determine areas of interest in a spectrogram and perturb those areas while observing the effect on the Siamese output. This approach is similar to the XAI LIME method applied to spectrogram features and Siamese networks. Our work uses a similar perturbation approach common to many XAI methods but has broader applicability across few-shot neural network architectures and conventional models.

3. The ProtoShotXAI Architecture

XAI ExMatchina uses a distance function at the feature layer which is similar to the distance metric used in few-shot algorithms such as Matching Networks and Prototypical Networks. The objective for both ExMatchina and the few-shot approaches is similar as well: to find exemplary sample(s) that are close in the network's feature manifold w.r.t the sample under test. The difference is that the few-shot architectures are designed to train a network for contrastive classification and ExMatchina is designed to explain an existing trained network. Specifically, few-shot trains a network to minimize (at the feature layer) the distance between a query sample and an exemplary prototype whereas ExMatchina simply finds (at the feature layer of a trained classification network) the nearest training sample w.r.t the query. ExMatchina does not use an exemplary prototype nor does it use information from the trained classification weights.

Our ProtoShotXAI method is an XAI architecture that uses both an exemplary prototype and the information from the classification weights (when available) to explain a trained model's decision. Specifically, ProtoShotXAI augments the feature layer with classification weights and computes a similarity score between an exemplary prototype and a query sample that needs explanation. Rather than finding the closest exemplary sample (as performed by ExMatchina), ProtoShotXAI computes an average exemplary prototype that has been shown to work well in few-shot learning research (Snell et al., 2017a; Vinyals et al., 2016). When used in few-shot explanations, ProtoShotXAI computes the average prototype at the feature layer; however, when used with conventional classification networks, our ProtoShotXAI approach combines the feature layer with the weights of the classification layer (i.e., just prior to feature summation and activation applied at the last layer) which we demonstrate improves explainability over state-of-the-art XAI approaches (see Section 4). We argue that, whereas methods like ExMatchina and Grad-SHAP remove layers of the classification network, ProtoShotXAI uses all the layers and is, therefore, more representative of the trained model and more applicable across different architectures.

After adding the few-shot architecture to a trained network, the similarity score can be used to provide explainable exemplary samples similar to ExMatchina. Additionally, the change in similarity score w.r.t. perturbations at the input provide insight into input feature importance and a feature attribution map can be produced (e.g., similar to the maps produced by LIME and SHAP). Unlike previous approaches, ProtoShotXAI provides the user with more flexibility to explore input feature changes and perturbations. For example, by using ProtoShotXAI on digits, the user can rotate a six and observe the change in similarity score w.r.t classes six and nine (see Section 4). Further, unlike LIME (which also

uses input perturbations), there is no separate model. The feature maps can be generated directly from the trained classification network.

An important distinction between ProtoShotXAI and ExMatchina is that ExMatchina is strictly an explain-by-example approach. ExMatchina does not provide the users with an analysis of the distance in the feature space as ProtoShotXAI can support. This opens up an important research question, namely, "*What would happen if we modified ExMatchina to analyze distances like we have done in our ProtoShotXAI approach*?" In this work, we make a modification to ExMatchina and entitled this new approach ExMatchina*. ExMatchina* combines the perturbation of our contributions with the original ExMatchina for comparison.

In Table 1, we summarize the key distinctions between our algorithms (ExMatchina* and ProtoShotXAI), and state-of-the-art XAI. Algorithms that provide evidence of counterfactuals indicate negative correlations between an input and an expected classifier output, which are not included in Grad-CAM, RISE, or ExMatchina. Concerning attribution weights on images, only RISE, LIME, and ProtoShotXAI provide full resolution maps (i.e., one-weight-to-one-pixel). Many of the approaches in Table 1 can be applied to any DNN classifier architecture; however, only LIME is known to be broadly applicable outside of DNN models; It is worth noting that few-shot is a special type of contrastive network that requires separate support and query sets and ProtoShotXAI is, to our knowledge, the only XAI model able to account for both in the XAI output.

The entire process of constructing the ProtoShotXAI architecture and use it to create feature attribution maps is shown in Figure 1. The ProtoShotXAI process starts with creating a trained classification network or using a preexisting trained network. Section 3.1 formally describes the process of constructing the ProtoShotXAI architecture from the baseline trained model and Section 3.2 describes one specific process for perturbing the data to produce feature attribution maps using ProtoShotXAI. The feature attribution maps (which can be directly compared against other approaches), as well as additional uses for ProtoShotXAI, are demonstrated in Section 4.

3.1 ProtoShotXAI Modification of a Trained Network

ProtoShotXAI modifies a pretrained classification network into the dual network architecture commonly used by many few-shot approaches such as Siamese networks, Matching networks, and Prototypical networks. That is, the trained DNN is replicated up to the feature layer for two branches: one branch contains the set of exemplary prototypes and the other branch contains variations of the query sample under observation. Formally, a classifier can be partitioned into a feature layer (the layer right before the classification layer) and a classification layer. The feature layer is represented by the nonlinear embedding function $\mathbf{f}(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^K$ which transforms a D-dimensional input sample $\mathbf{x} \in \mathbb{R}^D$ (such as an image) into a K-dimensional feature representation. At the classification layer, the features are multiplied by the trained static weights for each class node $\mathbf{w}^c$ where $c \in \{0, \ldots, C\}$ is a specific class out of the total number of class nodes C. The summation of the weighted features and bias b^c go through a softmax activation function for each of the classes and

	Deep SHAP	Expected Gradients	Grad-CAM	RISE	LIME	ExMatchina	ExMatchina* (Ours)	ProtoshotXAI (Ours)
Evidence of Counterfactuals	✓	✓			✓		✓	✓
Pairwise Comparisons						✓	✓	✓
Directly Applicable to Deep Few-Shot Networks								✓
Full Resolution Attribution Maps				✓	✓			✓
Deterministic	✓	✓	✓			✓	✓	✓
Model Agnostic					✓			
Agnostic to DNN Classifier Architecture				✓	✓	✓	✓	✓

Table 1: Tablular comparison of our algorithms (modified ExMatchina* and ProtoshotXAI) w.r.t. state-of-the-art methods evaulated throughout this manuscript.

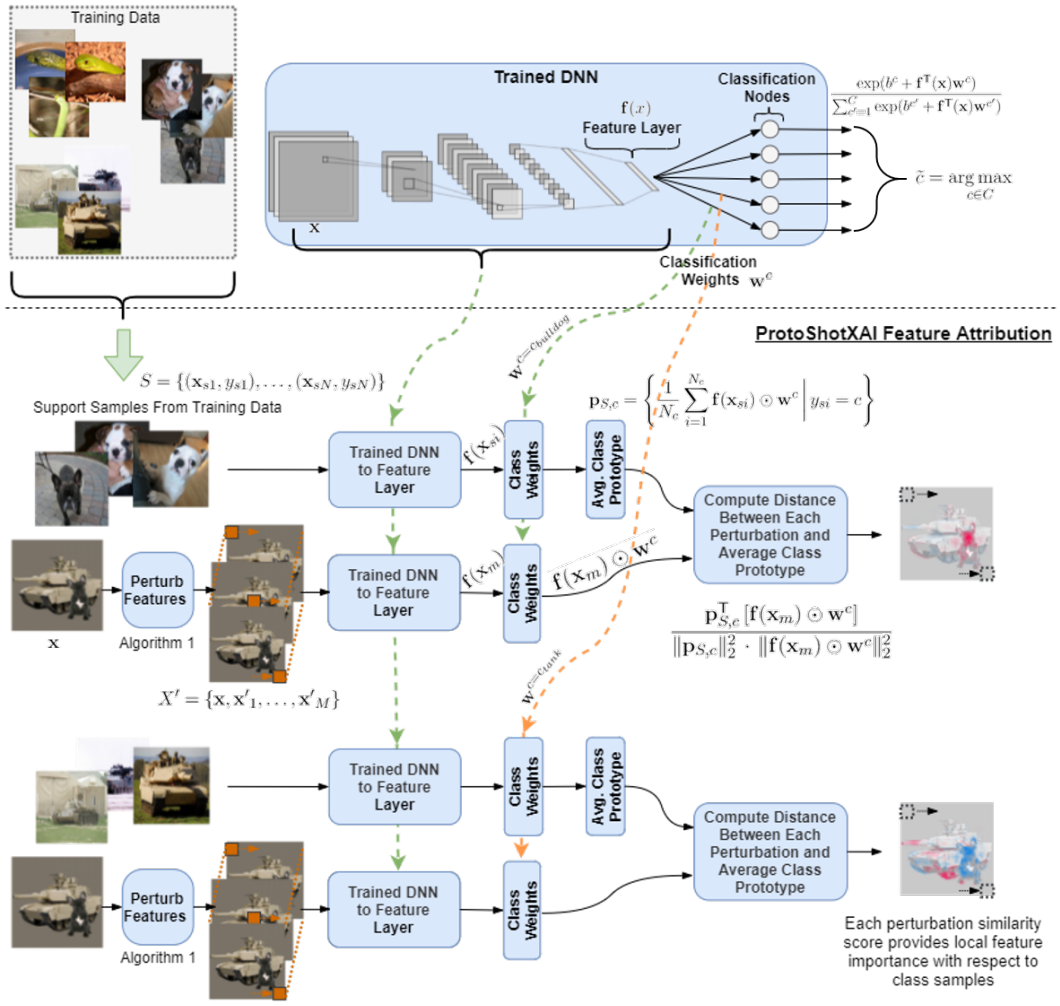


Figure 1: Overview of the architecture and feature attribution process of ProtoShotXAI. At the top of the figure is a trained DNN model and its corresponding training data. The ProtoShotXAI architecture has two branches. The first branch shows a set of support samples from the french bulldog passed through the trained DNN up to the feature layer and multiplied by the class weights of the french bulldog. A prototype vector is made from the average weighted features of the support samples. Similarly, the second branch contains the perturbations from the query sample (e.g., an image containing the tank and french bulldog). For each pixel perturbation image, a cosine distance is computed between the prototype vector and the weighted image features and the output is the feature attribution score at each pixel. The process done for the french bulldog support set is repeated for the tank support set to give positive and negative attribution scores w.r.t each class.

the estimated class $\tilde{c}$ is the argmax of the activation as given by the following

$$\tilde{c} = \arg\max_{c \in [C]} \frac{\exp(b^c + \mathbf{f}^\mathsf{T}(\mathbf{x})\mathbf{w}^c)}{\sum_{c'=1}^{C} \exp(b^{c'} + \mathbf{f}^\mathsf{T}(\mathbf{x})\mathbf{w}^{c'})} \tag{1}$$

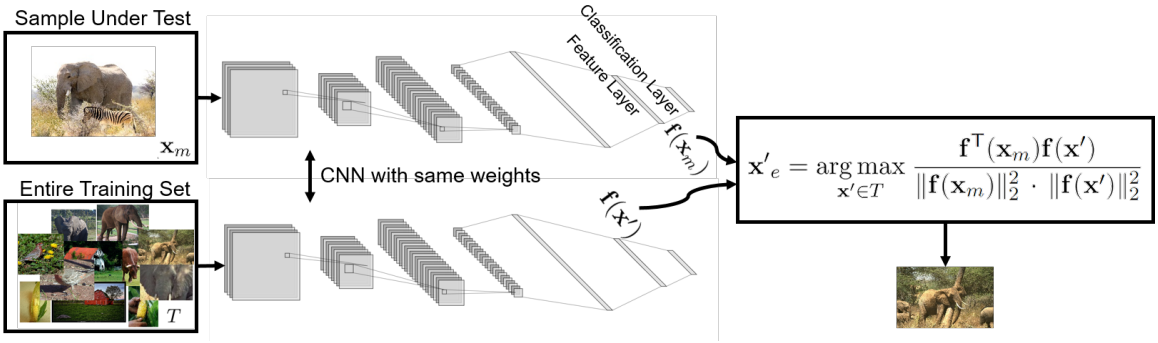
where $b^c \in \mathbb{R}$ is a scalar that represents the bias. As shown in the bottom of Figure 1, the first branch of the ProtoShotXAI architecture discards the activation, but keeps the trained network up until the feature layer and the weights of class. A subset $S = \{(\mathbf{x}_{s1}, y_{s1}), \ldots, (\mathbf{x}_{sN}, y_{sN})\}$ of support samples from a class are from the training dataset Tr (e.g., images the french bulldog class). This support subset can be randomly selected (as done throughout this work) or, if the user knows a set of representative exemplary samples, they can be hand picked. All the N D-dimensional support samples represented by $\mathbf{x}_{si} \in \mathbb{R}^D$ are passed through the DNN up until the feature layer and are multiplied by the weights of the specific class of interest (e.g., the trained weights from the french bulldog class). The prototype feature vector of the support set $\mathbf{p}_{S,c}$ is the average feature multiplied by the weights for class c.

$$\mathbf{p}_{S,c} = \left\{ \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{f}(\mathbf{x}_{si}) \odot \mathbf{w}^c \,\middle|\, y_{si} = c \right\} \tag{2}$$

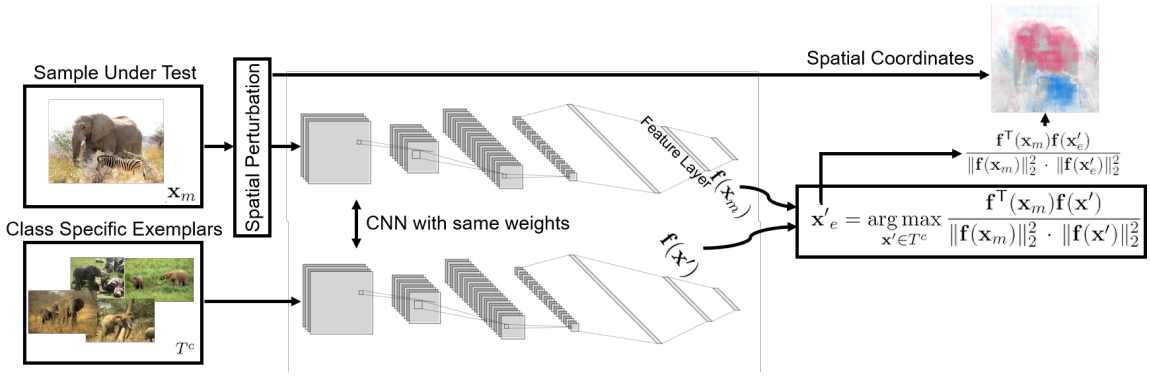
where $\odot$ is the Hadamard product (i.e., an element by element multiplication operation). For the second branch of ProtoShotXAI, the sample under test $\mathbf{x}$ (represented as the image containing both the tank and french bulldog in Figure 1) goes through a set of perturbations to observe the similarity response with respect to the prototype. Specifically, $\mathbf{x}$ is transformed into a perturbation dataset $X' = \{\mathbf{x}, \mathbf{x}'_1, \ldots, \mathbf{x}'_M\}$. This perturbed dataset can easily be created by applying small changes to a different pixel (or set of pixels) in each sample. Similar to the support samples, each $\mathbf{x}'_m$ goes through the DNN until the feature layer and is then multiplied by the weights of the specific class; however, no average is taken for the perturbations. Instead, each perturbation is compared against the prototype using a cosine similarity metric as represented in the following equation:

$$z_m(c, \mathbf{x}'_m) = \frac{\mathbf{p}_{S,c}^\mathsf{T} [\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c]}{\|\mathbf{p}_{S,c}\|_2^2 \cdot \|\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c\|_2^2} \tag{3}$$

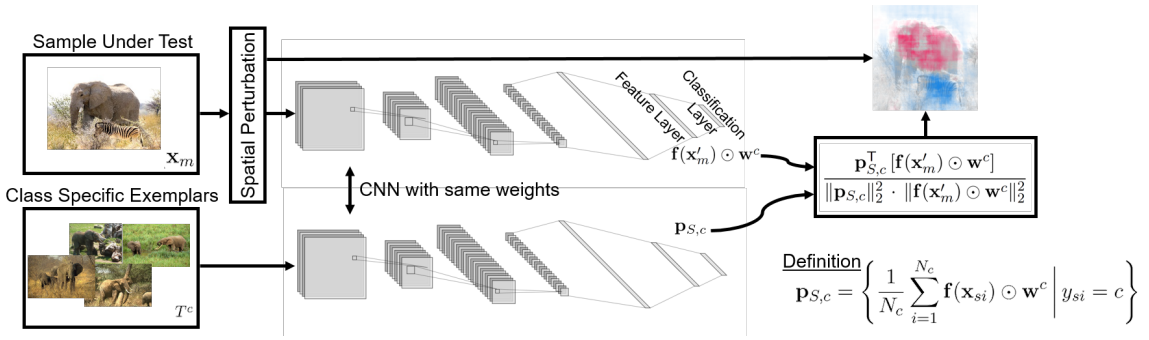
Figure 2 illustrates the difference between ExMatchina, ExMatchina* (ours) and ProtoShotXAI (ours). ExMatchina runs the sample under test and all the training samples through the same CNN and finds the most similar training sample (via the cosine similarity). ProtoShotXAI differs from ExMatchina in three distinct ways which are key contributions of our work for XAI: (1) the ProtoShotXAI distance function uses a prototype of features for its reference rather than the samples from a single feature, (2) ProtoShotXAI multiplies the features by the weights of a specific class node to get more representative features from the model, and (3) ProtoShotXAI uses input perturbations to cross-compare model responses to a baseline support set (like the feature attribution maps discussed in Section 3.2). In the case of few-shot approaches that already have a dual network architecture, the class weights can be set to a vector of ones and the same XAI approach can be used, making ProtoShotXAI more consistent across architectures.



(a) Exmatchina



(b) Exmatchina* (ours)



(c) ProtoShotXAI (ours)

Figure 2: Block diagram representations of Exmatchina, Exmatchina* (ours) and ProtoShotXAI (ours).

Algorithm 1 Implementation of ProtoShotXAI to Create Feature Attribution Maps

Input: Any representative set of N support image(s) $S = \{(\mathbf{x}_{s1}), \ldots, (\mathbf{x}_{sN})\}$ for a single class c, a image under test $\mathbf{x}$, average RGB pixel vector $\mathbf{R}_{RGB}$, a trained classification model with frozen parameters

Output: Feature attribution map $\mathbf{z}$.

$z_{ref} = \frac{\mathbf{p}_{S,c}^{\mathsf{T}}[\mathbf{f}(\mathbf{x}) \odot \mathbf{w}^c]}{\|\mathbf{p}_{S,c}\|_2^2 \cdot \|\mathbf{f}(\mathbf{x}) \odot \mathbf{w}^c\|_2^2}$ // Reference score of x w.r.t S

for $i = 1, \ldots, I$ **do** // Loop through columns of the image

  for $j = 1, \ldots, J$ **do** // Loop through rows of the image

    $m = j + (i-1)J$ // create a perturbation index for each 2D pixel

    $\mathbf{x}'_m = \mathbf{x}$ // $\mathbf{x}'_m$ is initialized as x before each perturbation

    $\mathbf{x}'_m(i,j) = \mathbf{R}_{RGB}$ // perturb the i,jth pixel to average RGB value

    $z(i,j) = z_{ref} - \frac{\mathbf{p}_{S,c}^{\mathsf{T}}[\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c]}{\|\mathbf{p}_{S,c}\|_2^2 \cdot \|\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c\|_2^2}$ // Score for $\mathbf{x}'_m$ w.r.t S

  end for

end for

3.2 Using ProtoShotXAI to Produce Feature Attribution Maps

After the construction of ProtoShotXAI, the entire architecture can be used in many ways to explore the feature space of the classification network. The ProtoShotXAI architecture receives two sets of inputs. One set are the support input features to compare against (e.g., samples from a single class) and the other can be any set of feature alterations to a query sample under test, such as adding perturbations, skewing features, adding noise, and/or setting features to zero. The change in the similarity score provides a measure for the effects of the feature alterations w.r.t to the class. That is, changing features that are more important to the network model will have a larger effect on the change to the similarity score.

As an example, input feature attribution maps can be created similar to the ones produced by LIME, SHAP, and Grad-CAM for images. We illustrate this implementation in the bottom of Figure 1 and present it at as pseudo-code in Algorithm 1. The algorithm starts by choosing a set of input support image(s) (e.g., images of the french bulldog in Figure 1) and image under test (e.g., an image containing both the tank and french bulldog in Figure 1). A support set can be created from a random sample(s) of a class in the training data or any representative set of exemplary sample(s) and the image under test is any image to be evaluated w.r.t. the support. After the support and query data are selected, the algorithm calculates a baseline similarity metric $z_{ref}(c, \mathbf{x})$ for reference. The algorithm then loops through each pixel in the image and replaces the RGB value of that pixel (or the small region surrounding that pixel) with an average value and computes the similarity metric for each spatial perturbation. Differences between the reference similarity score and the similarity score when the pixel is replaced are the feature attribution of that pixel. As the similarity score decreases, the pixel is weighted as being more important, and vice versa, as the similarity score increases.

4. Experiments

This section empirically demonstrates the proposed ProtoShotXAI approach on three benchmark datasets, namely: MNIST, Omniglot, and ImageNet. ProtoShotXAI is compared against several state-of-the-art XAI approaches, including ExMatchina, ExMatchina*, Grad-CAM, Grad-SHAP, RISE, and LIME. We describe multiple experiments where ProtoShotXAI can be used to explore the original classification network qualitatively and quantitatively, and demonstrate how our approach achieves comparable and often superior results to more complex algorithms for XAI. Throughout this section, we use **boldface** text when referring to characters. Specific details related to implementation, the datasets, and the neural network architectures is provided in Appendix A.

4.1 MNIST

In this section, we demonstrate three different aspects of ProtoShotXAI with experiments for the MNIST dataset. In the first experiment, we qualitatively evaluate the locally interpretable algorithms by visualizing the feature representations of the digit **eight**. **Eight** is a unique digit due to the subset of digits that are contained within it. For example, the digit **three** is commonly contained as a subset of **eight** as well as parts of **five**, **six**, and **nine**, making it convenient for cross-class explanations. In the second experiment, we demonstrate how ProtoShotXAI can be used to quantitatively assess the classification change as a **six** is rotated until it is classified as a **nine**. Finally, the last MNIST experiment analyzes the XAI approaches on an adversarial dataset. The adversarial dataset is unique because the true class and the predicted class from the classifier are often different and the features of ProtoShotXAI provide insight into the network's classification errors.

4.1.1 Feature Attribution on Eight

In the first experiment, we qualitatively evaluated ProtoShotXAI with an MNIST sample of the digit **eight**. The digit **eight** was chosen as a test sample due to the presence of multiple digits contained within the shape of the **eight**. The same **eight** was tested against each digit class (0-9) using the seven state-of-the-art XAI algorithms, namely, ProtoShotXAI, Deep SHAP, Grad-SHAP, Grad-CAM, LIME, RISE, and ExMatchina. The reference pixel value used in ProtoShotXAI is set to zero which indicates a lack of digit presence in a pixel. The results of each algorithm were normalized using the same standard applied in the SHAP work, which is necessary for a fair comparison, and the results are displayed in Figure 3.

One of the first observations from Figure 3 is that pixels with negative feature attributions are much more prevalent in ProtoShotXAI over the other approaches. The SHAP variants are the only other methods with negative feature attributions; however, the magnitude of those attributions are relatively small. The prominent negative features of ProtoShotXAI are attributed to the weights of the classification nodes which cause strong negative responses to counterfactual features of the given class (see Section 4.1.3 for more examples with adversarial perturbations). As a result, the remaining positive features in ProtoShotXAI highlight expected areas of interest. For example, the positive features for **two** are symmetrical to **five** about a vertical axis and the positive features for **six** are sym-

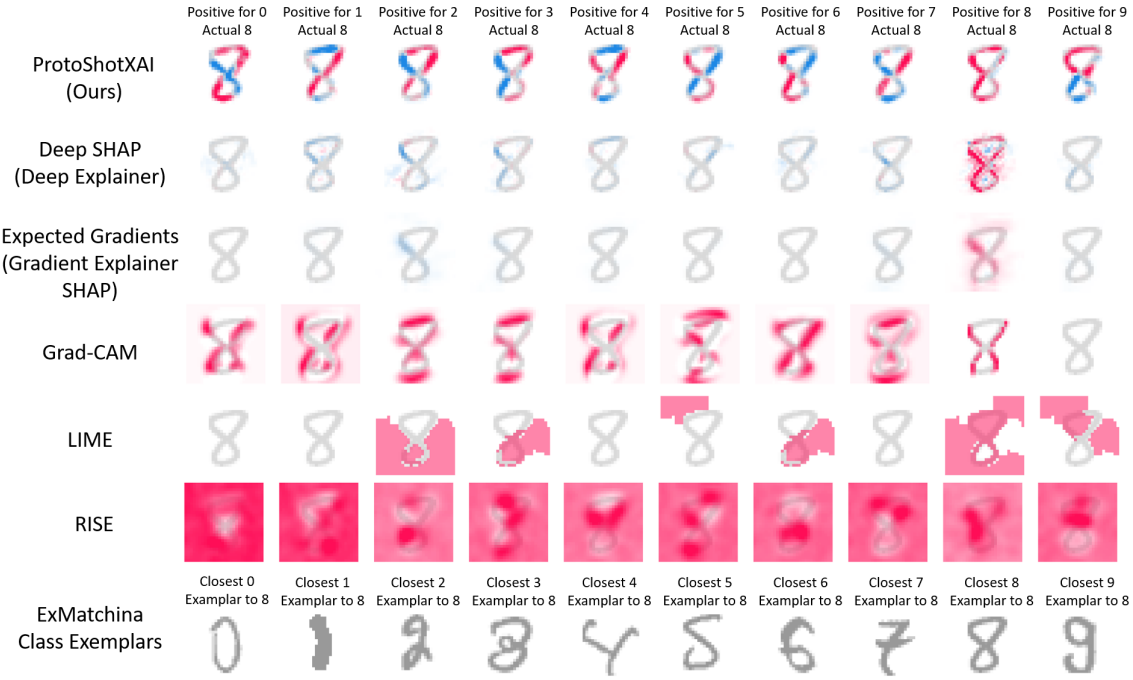


Figure 3: Example of feature attribution weights of various approaches on an exemplary character **eight** (shown in gray throughout). Consistent with the SHAP conventions, red areas indicate positive feature attributions and blue areas indicate negative feature attributions. ProtoShotXAI shows more contrast in features (e.g., positive attributes for digits **two**, **three**, **five**, and **seven** as sub-features of the digit **eight**) compared to other methods. ExMatchina is the one prototype method shown due to the mathematical similarity to ProtoShotXAI. ExMatchina only provides the nearest exemplary samples in a class which is apparent from provided exemplary digits like **three**, **five**, and **nine** but does not provide the same analytical clarity of feature attributions.

metrical to **nine** about a horizontal axis. Concerning the digit **three**, there is significantly more clarity in the ProtoShotXAI positive and negative features when compared to other methods. Additionally, ProtoShotXAI is more spatially precise than other methods. SHAP and Grad-CAM use only part of the convolution network layer and rely on spatial interpolation which causes a spatial blurring effect. LIME uses spatial segmentation areas that cause entire regions to be flagged as positive or negative feature attributes. ExMatchina is the only explain-by-example approach shown due to its mathematical similarity to ProtoShotXAI but it is not intended to provide cross-class comparisons. Hence, the closest samples ExMatchina provides are arguably "**eight**-like" (see exemplar **three**, **five**, and **nine**) but do not provide feature clarity like the other approaches in this context.

4.1.2 Revolving Six

We also performed a simple – yet meaningful – experiment that shows ProtoShotXAI can be used for more than feature attribution maps, the similarity score it produces can bring

fidelity to classification. Further, ProtoShotXAI has better agreement than the class of the nearest exemplary sample (i.e., ExMatchina). To demonstrate these properties of ProtoShotXAI, we consider an image of the digit **six** that is rotated from 0° to 360°. If a **six** is rotated 180° about the center of the digit, the image can be incorrectly classified as a **nine** by the network. It is worth mentioning that most humans would also incorrectly classify the rotated **six** as a **nine**. A question that arises from this scenario is: as a **six** is rotated from 0° and 180°, when does this transition occur that causes a different classification, and how rapid is the transition?

In this experiment, we rotate a **six** counterclockwise from 0° to 359° in increments of 1° then pass each rotated sample to a DNN. The model classifying these different rotational views of the **six** only predicts three different classes throughout the rotation: a **six**, a **zero**, and a **nine**. The transitions occur at 93° (transition from 6 to 0), 158° (transition from 0 to 9), 278° (transition from 9 to 0), and 329° (transition from 0 to 6). The top plot in Figure 4 shows the transition of the class predicted by the model (solid blue line), the ProtoShotXAI score (dashed red line), and the ExMatchina* score (dashed green line). For each of these rotated images, we use the ProtoShotXAI architecture to compute the distance between the rotated sample and a prototype of 100 random exemplary samples from each digit class of interest (i.e., classes **six**, **zero**, **five**, and **nine**). ProtoShotXAI scores for these classes of interest as the **six** is rotated are shown in the middle plot of Figure 4. For comparison, the same 100 random exemplary samples are used in ExMatchina* to retrieve the best cosine similarity within each class, which are shown in the bottom plot of Figure 4. Note that this evaluation of ExMatchina* was not proposed in the authors' original implementation. Since we revised ExMatchina from an explain-by-example approach to an analytical approach closer to ProtoShotXAI, we represent it as ExMatchina*.

The ProtoShotXAI and ExMatchina* scores have similar behaviors that are correlated to the model's predictions; however, these scores have better explainable fidelity. For example, the scores from ProtoShotXAI show a transition between classes of approximately 20°-30° before the model is certain about the next class. The difference between ProtoShotXAI and ExMatchina* in this synthetic example is that ProtoShotXAI finds the cosine similarity between the rotated **six** and the *average prototype* of the 100 exemplary samples, whereas ExMatchina* computes the cosine similarity between the rotated **six** and all the 100 exemplary samples then chooses the *prototype with the maximum similarity*. The result from ProtoShotXAI is more stable because the reference class prototypes will not change as the **six** rotates; however, this observation can not be said about the response of ExMatchina*'s score. The stability of ProtoShotXAI is further demonstrated by comparing the class predicted by the model (shown in the top plot of Figure 4) with the class that corresponds to the maximum score for ProtoShotXAI (dashed red line) and the maximum cosine similarity used in Exmatchina* (dashed green line). The output of the predicted class from ProtoShotXAI scores is more consistent with the class predicted by the model whereas the ExMatchina* cosine distance produces more deviations and even temporarily predicts a **five**. Therefore, we conclude ProtoShotXAI produces a similarity score that brings explainable fidelity to a model's prediction and provides more insight than ExMatchina*.

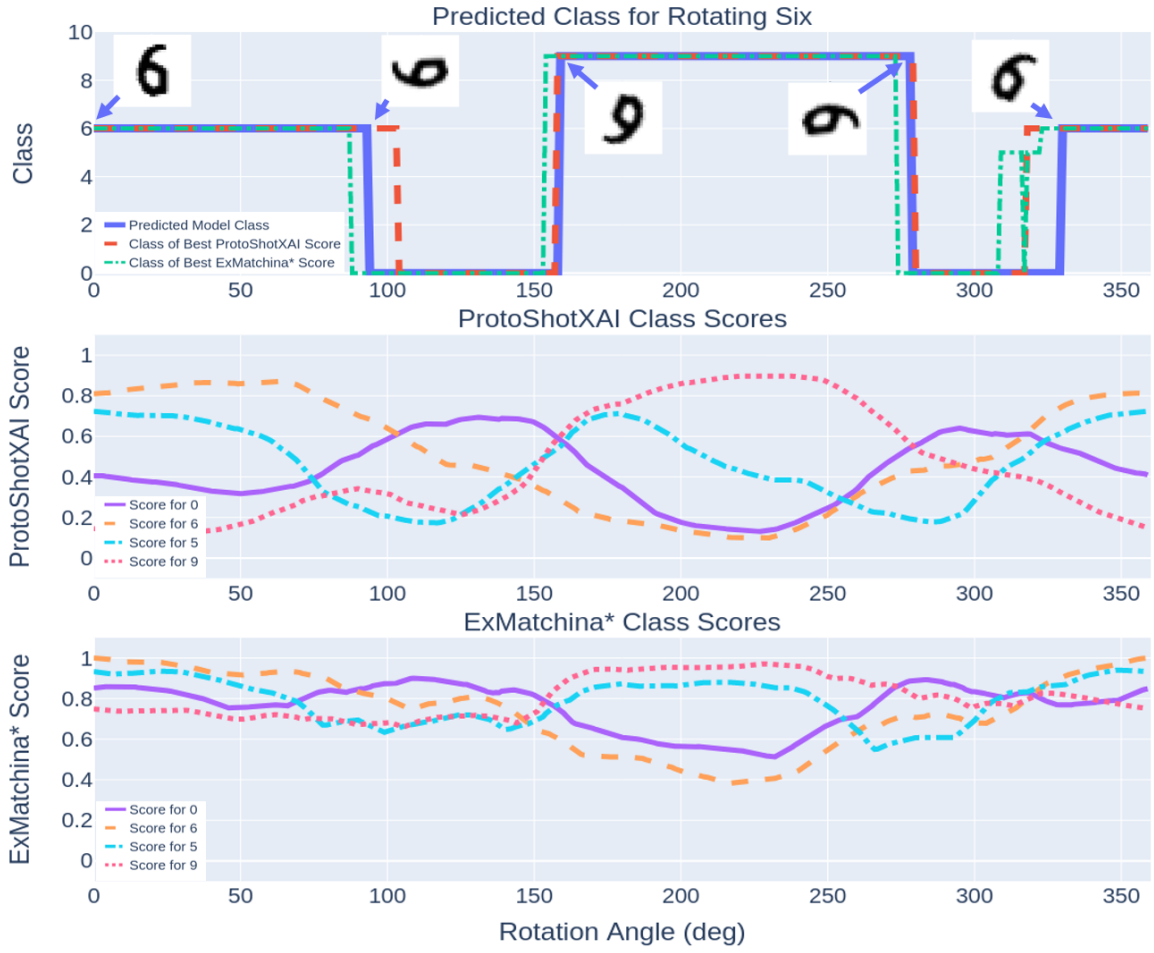


Figure 4: Predicted class as a **six** is rotated from 0° to 359° (blue line in the top plot) and corresponding ProtoShotXAI and EXMatchina* scores for classes **zero**, **six**, and **nine**. The red and green lines in the top plot result from assigning the class of the largest similarity score at each rotation angle for ProtoShotXAI and ExMatchina*, respectively. ProtoShotXAI has more consistency with the model predictions due to the prototype averaging.

4.1.3 Adversarial MNIST

In the context of classification, adversarial data are crafted to exploit a model's behavior to produce incorrect predictions by adding a small perturbation to the original image that is imperceivable to a human but fools the model (Goodfellow et al., 2014; Sadeghi et al., 2020; Qiu et al., 2021). For example, specific small perturbations to an MNIST digit can appear to be minor alterations to the human eye yet cause drastic changes to the output prediction of a trained neural network. This behavior causes hesitancy in integrating neural networks in precarious situations where an adversary might be able to inflict malicious data with the intent of causing deleterious predictions. As a result, a relevant goal of XAI is to explain when and why adversarial data corrupts model behavior. If XAI can explain when and why, then deleterious predictions from an adversary can be bounded or mitigated which increases user trust in the black-box model.

To explore the adversarial behavior of ProtoShotXAI, we crafted an adversarial dataset by taking the trained neural network from Table 6 and training it on the MNIST dataset. The adversarial samples are generated from the testing dataset using the *Fast Gradient Sign Method* (FSGM), which generates adversarial samples of the form (Goodfellow et al., 2014):

$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \,\text{sign}(\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y)) \tag{4}$$

where $\epsilon > 0$ is the adversarial budget, θ are the parameters of the network and $L(\cdot, \cdot, \cdot)$ is the categorical cross-entropy loss. The adversarial samples are presented to the ProtoShotXAI network.

The results of this experiment are shown in Figures 5 and 6. There are several observations that we make by considering the results in Figure 5. The feature attribution maps of all methods on the adversarial data were noisy and provided little-to-no insight; however, by plotting the component weights of the support prototype, the query sample, and the ProtoShotXAI score, we noticed a clear difference between the adversarial samples and benign samples. The components are the raw element values that make a vector. For example, in Figure 5 the components for the support set are the values of the vector $\mathbf{p}_{S,c}$ (as defined by equation 2), the query set are the values of the vector $\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c$, and the ProtoShotXAI components are the values of the vector $\mathbf{p}_{S,c} \odot [\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c]$ (as defined in the numerator of equation 3 prior to the summation of components).

In this experiment, a benign **four** and adversarial **four** were selected from the testing dataset to demonstrate this difference. As shown in the top two plots of Figure 5, a benign **four** correlates well to the support prototype of a **four** (top left plot), which causes prominent peaks in the ProtoShotXAI components and subsequent scores. The prototype of a **five** causes the query **four** to produce a lot of negative weights and is not correlated well to the support prototype of a **five** (top right plot). The result is as expected: the benign **four**'s score for the class **four** prototype is much larger than the class **five** prototype. However, in the adversarial query sample of a **four**, neither the score for the prototype of the **four** nor the prototype of the **five** are well correlated (bottom left and right plots, respectively). The overall score is poor for both classes but is marginally worse for class **four** than for class **five**. As a result, the adversarial sample is incorrectly classified as the **five** class.

The variation between the adversarial and benign scores for the in-class samples in Figure 5 is novel to the ProtoShotXAI approach and begs the question: *can ProtoShotXAI be useful*

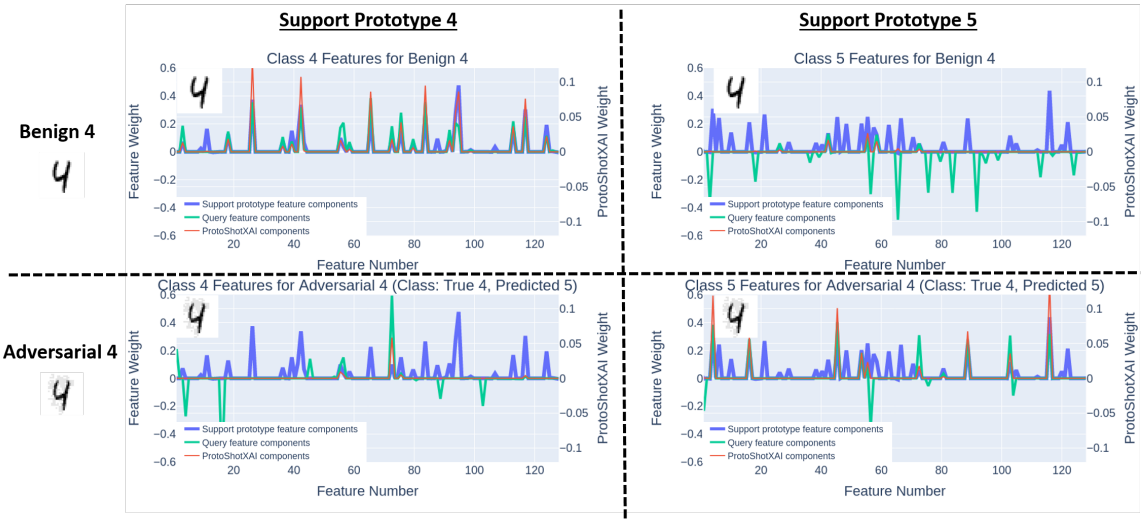


Figure 5: Results on the adversarial MNIST dataset which shows the decomposition of feature components from a benign **four** (top) and adversarial **four** (bottom) w.r.t. class **four** prototype (left) and class **five** prototype (right). The components for the support set are the values of the vector $\mathbf{p}_{S,c}$ (as defined by Eq. (2)), the query set are the values of the vector $\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c$, and the ProtoShotXAI components are the values of the vector $\mathbf{p}_{S,c} \odot [\mathbf{f}(\mathbf{x}'_m) \odot \mathbf{w}^c]$ (as defined in the numerator of Eq. (3) before the summation of components). From these plots, the benign **four** has a very positive response to the class **four** prototype and a negative response to the class **five** prototype. The adversarial samples do not show this same behavior, it is more uniformly distributed across features. ProtoShotXAI can demonstrate at this level that the features are corrupted, but what components caused feature corruption is still unanswered.

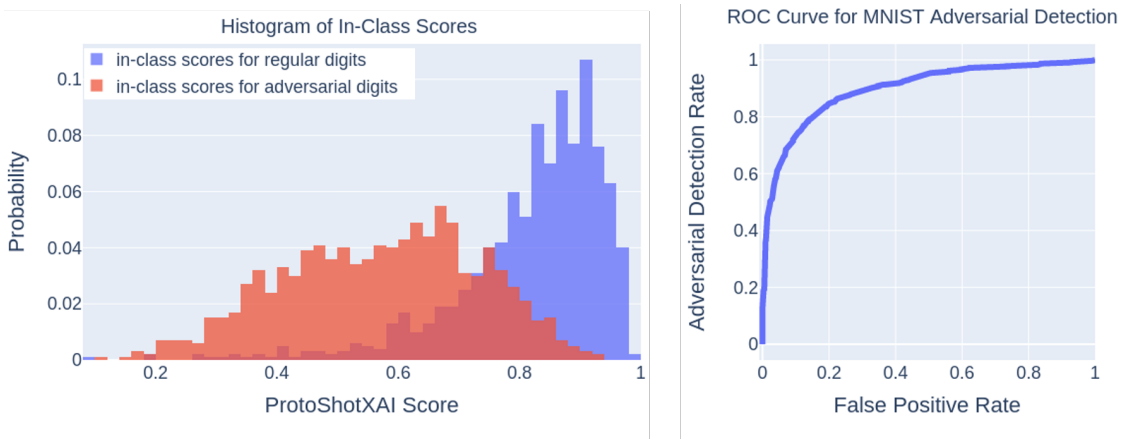


Figure 6: The plot on the left shows the histogram of the in-class scores for the benign **four** (in blue), the adversarial **four** (in orange), and the overlap (in red). The plot on the right is the ROC curve for a simple histogram threshold detector applied to the ProtoShotXAI scores. The result is the ProtoShotXAI scores can be used to detect adversarial samples by knowing the in-class scores from a validation dataset.

to identify adversarial samples? From the analysis of ProtoShotXAI scores, the results show that the adversarial samples corrupt the network at the feature layer by decorrelating the feature weights from the classification weights. The features in a benign sample strongly align to the weights of its own class, whereas the adversarial features do not strongly align with any class. This lack of alignment to a class causes minor correlations to decide the prediction. This result is made clear by running an experiment that generates the in-class ProtoShotXAI scores for 1,000 random samples of both the benign and adversarial MNIST data. The benign and adversarial score distributions are shown in the left plot of Figure 6. As expected, the adversarial digits have a significantly lower average than the average ProtoShotXAI scores on benign digits. It is also worth noting that the variance of ProtoShotXAI's score is larger than its benign counterpart. A simple detector can be created by applying a threshold to the ProtoShotXAI score. The right plot of Figure 6 shows the receiver operating characteristic (ROC) curve for adversarial detection versus falsely detecting benign digits as adversaries. The results of the simple threshold detector demonstrate that ProtoShotXAI can answer when a sample is adversarial and helps to improve model trust; however, it does not directly answer why the sample is adversarial (i.e., why from an interpretability perspective). We leave the analysis of "why" the sample is adversarial as future work and this question of "why" is an ongoing area of research (Merrer and Trédan, 2020; Tomsett et al., 2020). Further, the FGSM attack was the only perturbation generation approach evaluated, which is one of the simplest attacks to produce. Therefore, we leave a study of more complicated and difficult to defend against attacks to future work (Tramér et al., 2017; Li and Ditzler, 2022).

(a) Omniglot character (b) Feature explanations

Figure 7: An example of an Omniglot character (left) and ProtoShotXAI feature attributions w.r.t in-class prototype using Algorithm 1 (right). The features indicate the strongest weighting to the sub-character on the top left and the weakest weighting to the sub-character on the top right. We demonstrate a similar conclusion when looking at the similarity score w.r.t removing the larger sub-characters as shown in Figure 8.

4.2 Omniglot

ProtoShotXAI borrows from a few-shot architecture which makes it easy to apply to few-shot classification methods. To the best of our knowledge, ProtoShotXAI is the first method to explicitly apply XAI to datasets such as Omniglot. Since many few-shot architectures use the dual network structure for support and query samples, applying ProtoShotXAI to few-shot tasks is equivalent to a standard neural network with the classification weights set to a vector of ones. Figure 7 shows a character from the Omniglot database (left) that can be classified with the trained model from Table 7. Similar to MNIST, we used ProtoShotXAI with Algorithm 1 and a reference pixel value of zero which indicates a lack of digit presence in a pixel. Although many pixels have a positive in-class response for the Omniglot character (right), it is observed that the order of importance appears to be the top left, the bottom, and the top right sub-character. Additionally, the specific size of the top right sub-character appears to be less important than the spatial center based on the strong positive response from the sub-character's spatial center.

In contrast to feature attributions, ProtoShotXAI can evaluate the response to large-scale feature changes. Using the previous character as an example, we removed each of the respective sub-characters to determine the overall network response to its class using the ProtoShotXAI score. Figure 8 shows the resultant scores on a line plot. The overall score dropped the most by removing the top left sub-character, indicating that it was most important to the overall character classification. Conversely, removing the top right sub-character caused little change to the overall score, indicating that it was least important to the overall character classification. This result corresponds to the same order of sub-character importance obtained from the feature attribution maps.

4.3 ImageNet

The previous sections evaluated ProtoShotXAI against state-of-the-art digit classification methods (MNIST) and also demonstrated that ProtoShotXAI could be used for few-shot character classification (Omniglot). Both the MNIST and Omniglot datasets are images

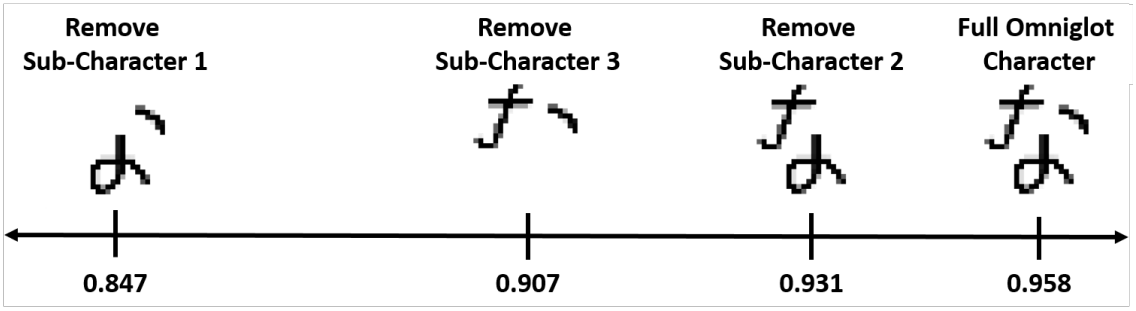


Figure 8: ProtoShotXAI similarity score w.r.t in-class prototype. The image on the far right with the highest similarity score corresponds to the full Omniglot character where the three images on the left correspond to the ProtoShotXAI score with different sub-characters removed. Removing the top left sub-character causes the largest negative deviation from the similarity score, indicating it is the most important sub-character of the three. The order of sub-character importance agrees with the feature attribution maps shown in Figure 7.

with approximately binary pixels; however, many real-world applications have complex features or pixels. The experiments in this section aim to qualitatively demonstrate that the ProtoShotXAI approach can be applied to datasets with complex image features, namely ImageNet. In the first example (see Figure 9), we reduce contextual clues by creating an artificial input of an image containing two unrelated classes within the 1,000 ImageNet classes: a tank and a french bulldog. The background for this image is an average pixel value that appears as a brownish color and provides no contextual information. In Figures 10 and 11, we show two more examples of real images that contain both contextual background and two class objects in a single image. Figure 10 contains an elephant and a zebra, whereas Figure 11 contains a cat and a dog. For each image, we show the attributions produced from a total of four pretrained classification networks used in the assessment: VGG16, Xception Network, and ResNet50, all with ImageNets weights, and ResNet50 with Stylized-ImageNet weights. "ImageNet weights" refer to the default weights provided by Keras and the "Stylized-Imagenet weights" refer to weights from Stylized-Imagenet training as discussed in Appendix A. Stylized-ImageNet substitutes textures in the images with the goal of training a network that is biased towards shape instead of texture. All three figures show the comparison of ProtoShotXAI, Grad-SHAP (SHAP Gradient Explainer), Grad-CAM, LIME, and RISE for each trained network. Only LIME, RISE, and ProtoShotXAI were applicable to Xception Network due to the network's complexity. The gradient-based methods (i.e., Grad-SHAP and Grad-CAM) operate at a specific convolutional layer and break down with unconventional convolutional architectures (like that of Xception Network). In contrast, LIME and ProtoShotXAI use the full model and do not have the same limitation. In this work, Grad-CAM operates at the last convolutional layer and Grad-SHAP operates on layers 7 and 38 for the VGG16 and ResNet architectures, respectively. The choice of the convolutional layer is a hyperparameter in Grad-SHAP where deeper layers tend to provide a better representation at the cost of lower spatial resolution. The layer was selected based on a comparable spatial resolution to the other methods, which results in a fair comparison.

For each trained network and each example image, ProtoShotXAI demonstrates consistent positive features (red) on the class of interest and negative features (blue) on the counter class. For example, ProtoShotXAI demonstrates consistent positive features for the tank attributions around barrel/treads and negative features in the region of the french bulldog (see Figure 9). For the french bulldog attributions, ProtoShotXAI has positive responses around the face and negative responses scattered throughout the tank. All methods appear to have inconsistencies w.r.t attribution and class (e.g., some positive feature attributions on the tank for the french bulldog class); however, the response of ProtoShotXAI is more spatially consistent throughout. Grad-SHAP is the closest to ProtoShotXAI for the VGG16 class with more "incorrect" positive features on the tank for the french bulldog class. Additionally, Expected Gradient's dot-like appearance on ResNet50 is from the method's use of deeper convolutional layers that partition the image into low-resolution spatial features. In contrast, the other gradient method, Grad-CAM, has a more blurred appearance and only identifies positive features of interest. The blurred appearance causes background regions to be positively indicated as feature attributions as well. LIME does not have the same issue of positively indicated feature attributions due to a segmentation algorithm that causes very defined region edges; however, LIME also segments large regions inconsistently, such as the positive feature attributions around the french bulldog's feet w.r.t the tank class.

ResNet50 with Stylized-ImageNet is unique to this experiment because it is the only network trained with the modified ImageNet dataset meant to remove the network's texture bias while maintaining the shape bias. We expected to see different network responses that indicate isolation of feature boundaries rather than regions, and unfortunately, this behavior was not explicitly observed. The feature attribution maps for ResNet50 trained with Stylized-ImageNet are drastically different from ResNet50 trained with the original ImageNet but it is inconclusive to argue that any XAI feature attribution methods show evidence of a shape bias. Although considered out of scope for this work, our XAI results warrant further investigation to determine if ResNet50 (trained with Stylized-ImageNet) has learned some complex features other than explicit shapes, such as the underlying textures from the stylized transformation.

4.4 Insertion and Deletion

In this section, we replicate Petsiuk et al. (2018)'s insertion and deletion experiments for LIME, Grad-CAM, Grad-SHAP, ProtoShotXAI and RISE. Specifically, we randomly select 250 samples from ImageNet's validation set and run each explainer with the four networks (i.e., VGG16, ResNet50 w/ImageNet Weights, ResNet 50 w/Stylized ImageNet weights, and Xception). We performed the deletion to a zero canvas and insertion to a blurred canvas as proposed in Petsiuk et al. (2018) and the pixel step size was set to 32. Table 2 shows our findings for the area under the curve (AUC) metric computed in Petsiuk et al. (2018). Unfortunately, when the results are carefully investigated, they are inconclusive. First, given standard error measurements, many approaches are not statistically different enough to warrant a clear ranking for any column. Second, no clear XAI approach performs statistically better than the others. For example, SHAP appears to do the best for deletions; however, the approach often does the worst at insertions.

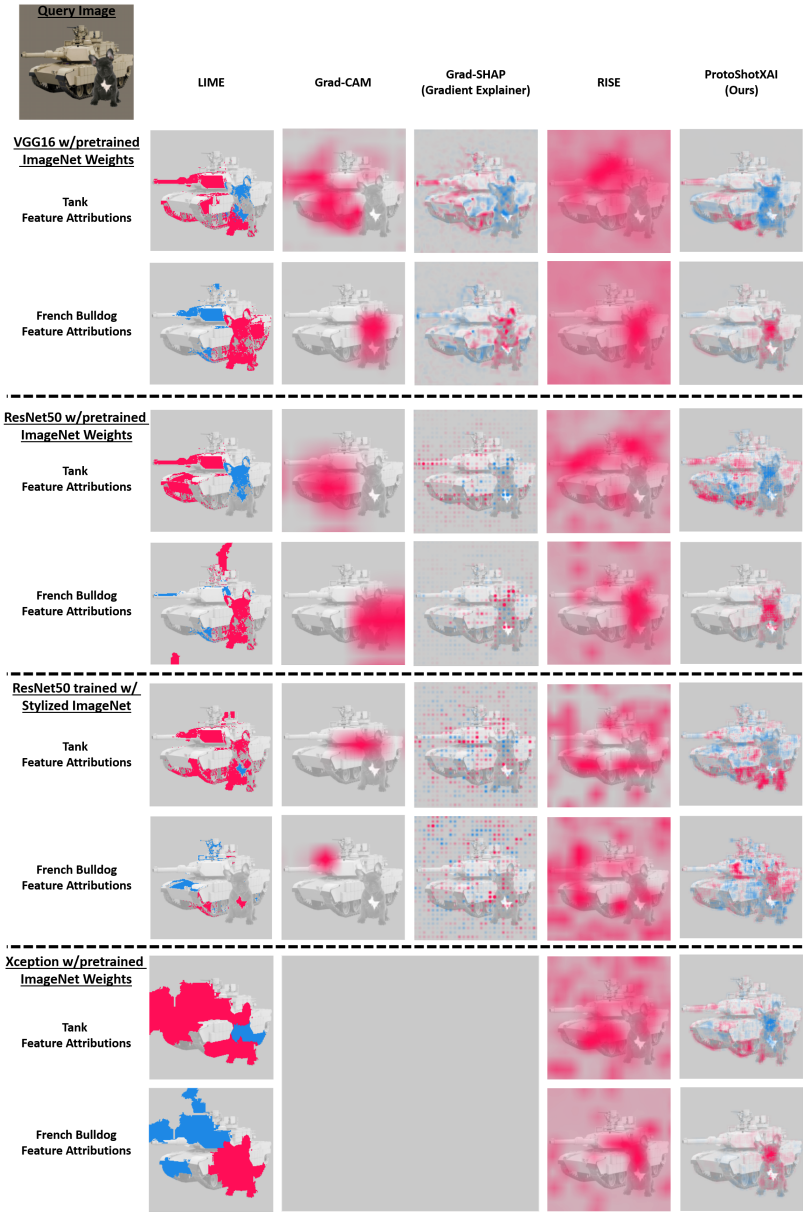


Figure 9: Comparison of feature attributions produced by five different methods (LIME, Grad-CAM, Grad-SHAP, RISE, and ProtoShotXAI) and four different networks. The query image containing the tank and french bulldog is shown in the top left. Each row corresponds to a feature attribution of a given network for a specific class (i.e., tank or french bulldog class). Each column corresponds to a particular approach. Due to the complexity of the Xception Network architecture, the gradient methods, which expect a specific convolutional architecture, were not applicable. The feature attributions from ProtoShotXAI are noticeably more consistent (i.e., positive and negative feature attributions are more isolated to their respective class region).

Figure 10: Comparison of feature attributions on a real image of an African elephant and a zebra. The query example is shown in the top left. Similar to the findings in Figure 9, ProtoShotXAI attributions are noticeably more consistent (i.e., positive and negative feature attributions are more isolated to their respective class region).

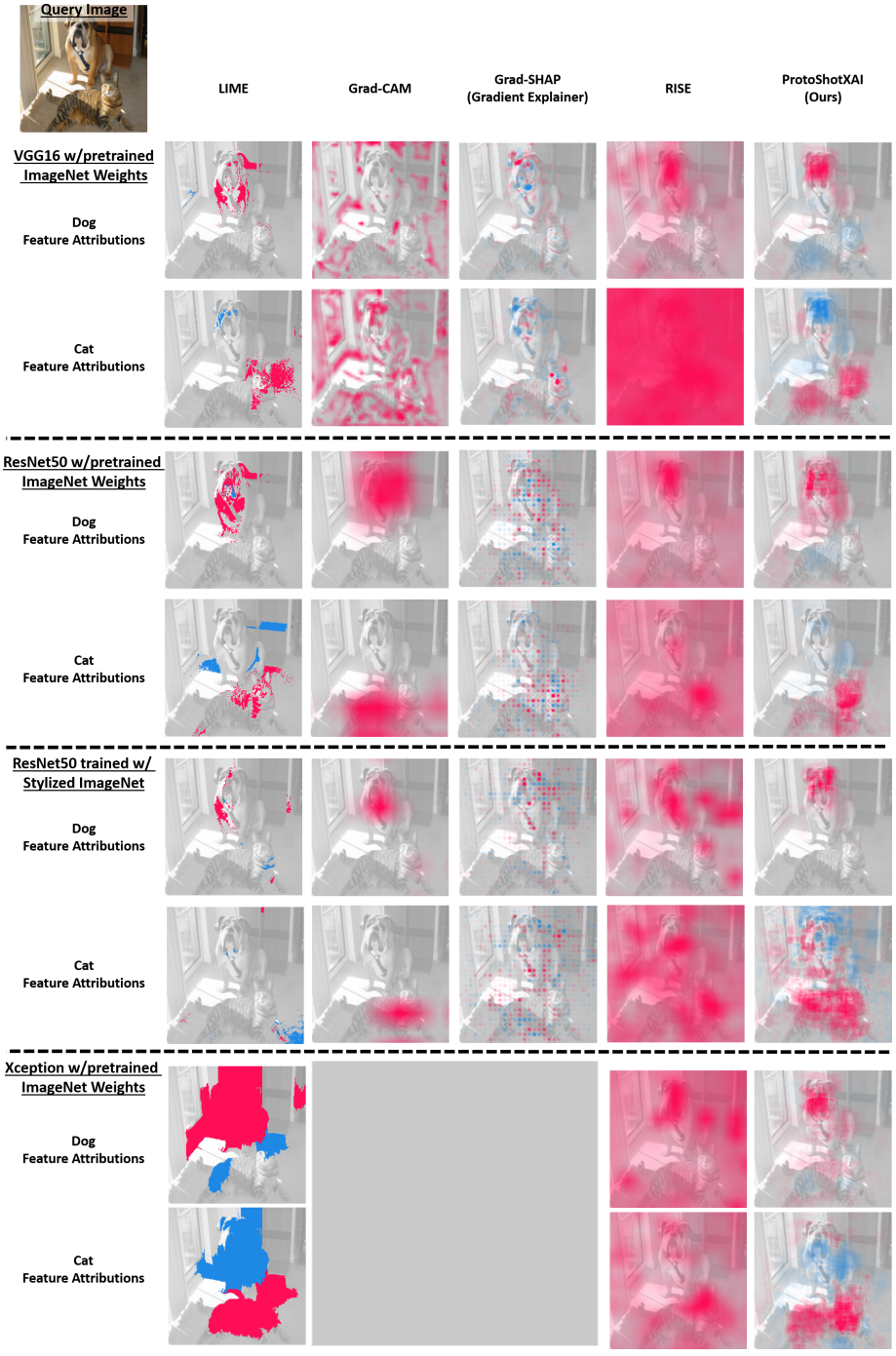


Figure 11: Another comparison of feature attributions on a real image. The query example of a cat and dog is shown in the top left. Similar to the findings in Figure 9 and 10, ProtoShotXAI attributions are noticeably more consistent (i.e., positive and negative feature attributions are more isolated to their respective class region).

Table 2: Deletion/Insertion AUC metric

Model	Network Model: VGG16		ResNet50	
	Deletion	Insertion	Deletion	Insertion
LIME	0.0651 ± 0.0071	0.3326 ± 0.0187	0.0705 ± 0.0071	0.4113 ± 0.0209
Grad-CAM	0.0707 ± 0.0075	0.2820 ± 0.0161	0.0999 ± 0.0082	0.5340 ± 0.0222
Grad-SHAP	0.0288 ± 0.0029	0.4676 ± 0.0206	0.0214 ± 0.0037	0.4050 ± 0.0187
RISE	0.0687 ± 0.0061	0.4745 ± 0.0221	0.0823 ± 0.0069	0.5451 ± 0.0222
ProtoshotXAI (Ours)	0.0565 ± 0.0060	0.4495 ± 0.0208	0.0806 ± 0.0084	0.5111 ± 0.0214
Model	**Network Model: Stylized ResNet50**		**Xception**	
	Deletion	Insertion	Deletion	Insertion
LIME	0.0419 ± 0.0064	0.2053 ± 0.0189	0.0631 ± 0.0061	0.5882 ± 0.0207
Grad-CAM	0.0452 ± 0.0065	0.2118 ± 0.0183	NA	NA
Grad-SHAP	0.0306 ± 0.0035	0.2565 ± 0.0192	NA	NA
RISE	0.0302 ± 0.0038	0.2440 ± 0.0196	0.0963 ± 0.0080	0.6032 ± 0.0233
ProtoshotXAI (Ours)	0.0244 ± 0.0042	0.2203 ± 0.0182	0.0967 ± 0.0101	0.5863 ± 0.0219

We present two representative examples of the variability of the deletions. In Figure 12, SHAP does extremely well on deletions in the turtle image; however, SHAP performs poorly on the explanations of the clock (see Figure 13). We often found this observation when analyzing these plots on the evaluation dataset. One of the primary takeaways from this experiment is that no one method is universally better w.r.t. all metrics. Table 3 breaks down the percentage each algorithm "won" the AUC (i.e., lowest in deletions or highest in insertions for each image). We observe there are some discrepancies between the algorithm's number of "wins," and which algorithm achieves the average AUC metric in Table 2. For example, RISE and Grad-CAM have an average AUC for ResNet50 Insertion of 0.5451 and 0.5340, respectively. Although RISE's AUC is slightly higher, Grad-CAM has a higher "win" rate of 38% compared to RISE's "win" rate of 18.8%.

We additionally note that while Grad-SHAP appears to perform better than other algorithms w.r.t deletions, the benefits come from the polka-dot pattern (e.g., see the Grad-SHAP image in Figure 12). To demonstrate this result, we replicate the polka-dot pattern in the other methods by forming a grid of ones and zeros with the shape 56×56 then upsampling the grid to 224×224. This upsampled pattern is used to weight the attribution maps (i.e., one for LIME, Grad-CAM, RISE, and ProtoShotXAI). Then the insertion and deletion experiment is rerun on ResNet50. The results are shown in Table 4. We found the polka-dot weighting improves the deletion performance of all the other methods that we evaluated.

4.4.1 Summary of Computational Resources

In Table 5, we have included the run times to produce the batch of 250 explanations in this experiment. The algorithms were run on a Lambda Labs node with 56 CPUs, 400 GB RAM, and four Nvidia A6000 GPUs with 48 GB of RAM each. The results

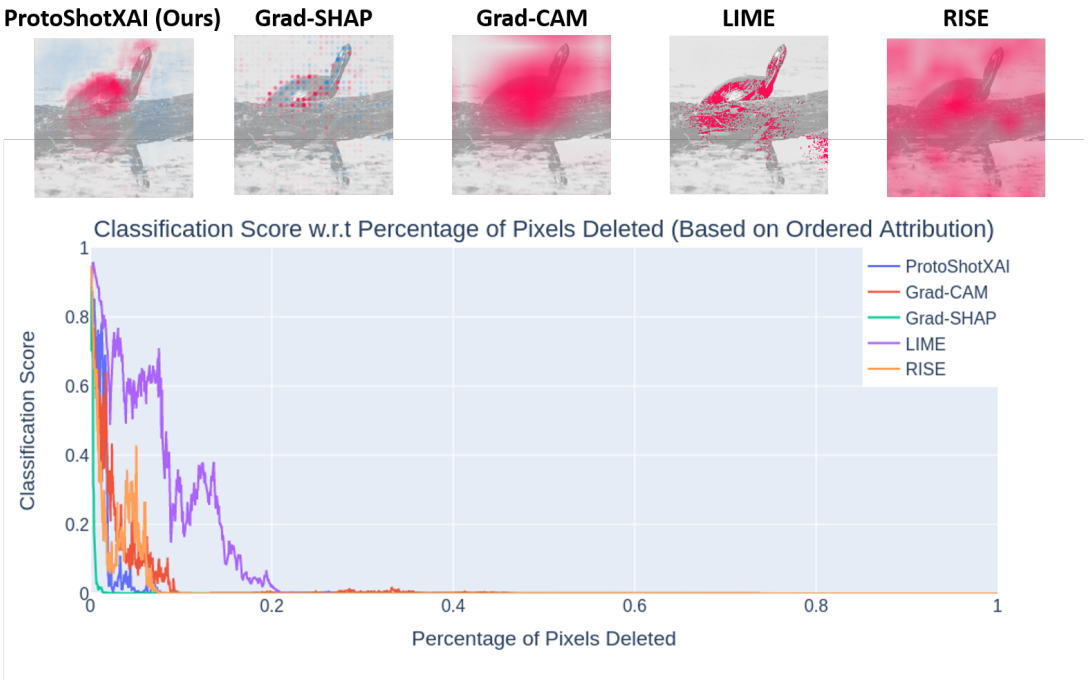


Figure 12: The classification score of a turtle as a function of the percentage of pixels "removed" from the image. The pixels are removed by the sorted order of the pixel attribution obtained from each method. The highest attribution pixel is "removed" first where "removed" is with respect to a black canvas (i.e., on the far right of the plot each image converges to a completely black image). The area under the curve is the metric of interest reported where the smaller areas (e.g., Grad-SHAP, ProtoShotXAI) are considered "better" than larger areas (e.g., LIME).

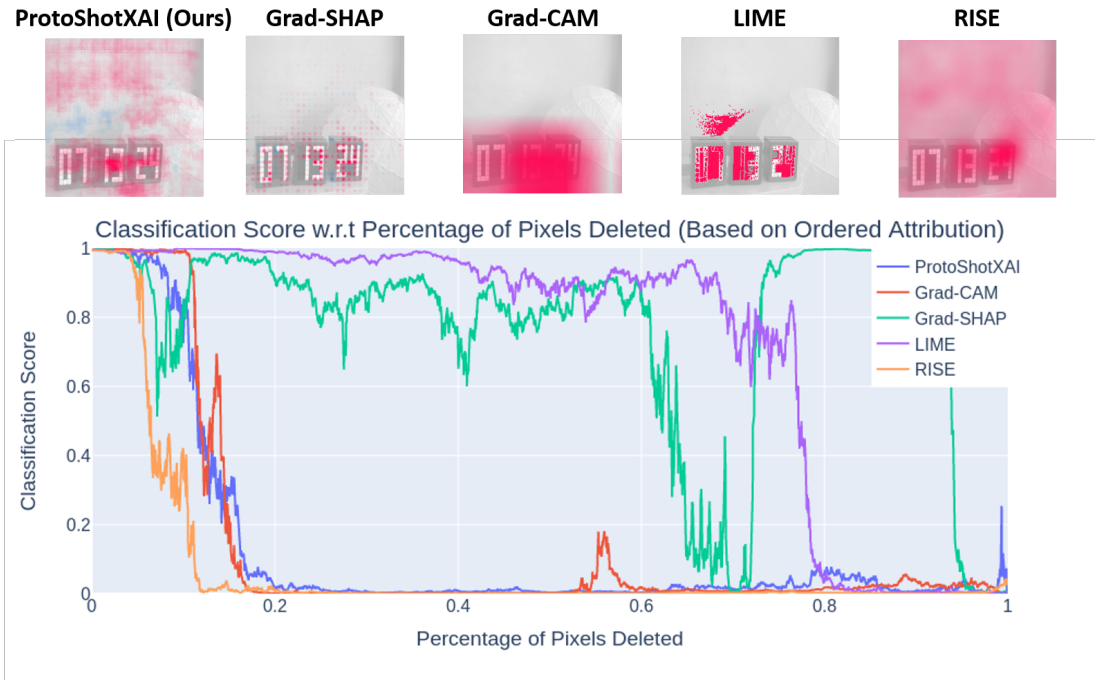


Figure 13: Similar to Figure 12 except for the classification of a clock image. In contrast to Figure 12, Grad-SHAP has one of the larger areas under the curve, and RISE has the smallest area under the curve. This indicates that the average area under the curve may not be a universal metric for performance

Table 3: Deletion/Insertion % Win Metric

Model	**Network Model**			
	VGG16		**ResNet50**	
	Deletion	Insertion	Deletion	Insertion
LIME	19.6%	23.6%	14.4%	21.6%
Grad-CAM	9.6%	28.0%	2.4%	38.0%
Grad-SHAP	42.0%	4.0%	74.4%	3.2%
RISE	10.4%	32.4%	5.2%	18.8%
ProtoshotXAI (Ours)	18.4%	12.0%	3.6%	18.4%
Model	**Network Model**			
	Stylized ResNet50		**Xception**	
	Deletion	Insertion	Deletion	Insertion
LIME	19.6%	26.4%	46.8%	25.6%
Grad-CAM	19.2%	18.0%	NA	NA
Grad-SHAP	14.4%	25.2%	NA	NA
RISE	19.2%	22.0%	22.4%	43.6%
ProtoshotXAI (Ours)	27.6%	8.4%	30.8%	30.8%

Table 4: Deletion/Insertion AUC Metric on ResNet50 with and w/o Grid Weighting

Model	**Network Model**			
	w/o weighting		**with weighting**	
	Deletion	Insertion	Deletion	Insertion
LIME	0.0705 ± 0.0071	0.4113 ± 0.0209	0.0672 ± 0.0071	0.4166 ± 0.0211
Grad-CAM	0.0999 ± 0.0082	0.5340 ± 0.0222	0.0328 ± 0.0034	0.4874 ± 0.0212
Grad-SHAP	0.0214 ± 0.0037	0.4050 ± 0.0187	0.0214 ± 0.0037	0.4050 ± 0.0187
RISE	0.0823 ± 0.0069	0.5451 ± 0.0222	0.0224 ± 0.0030	0.3978 ± 0.0201
ProtoshotXAI (Ours)	0.0967 ± 0.0101	0.5863 ± 0.0219	0.0573 ± 0.0067	0.4776 ± 0.0209

Table 5: Run time for a batch of 250 explanations

Model	**Network Model**			
	ResNet50	Stylized ResNet50	VGG16	Xception
LIME	16.79 hours	16.49 hours	7.32 hours	19.21 hours
Grad-CAM	53 seconds	58 seconds	27 seconds	–
Grad-SHAP	6.61 hours	6.26 hours	5.82 minutes	–
RISE	6.38 hours	6.28 hours	5.28 hours	5.77 hours
ProtoshotXAI (Ours)	12.13 hours	12.16 hours	17.13 hours	11.32 hours

show that Grad-CAM is exceptionally fast w.r.t. all other algorithms, producing all 250 explanations in under a minute. All other algorithms completed the 250 explanations on the scale of several hours. Grad-SHAP runs considerably faster on VGG16 than ResNet50. The exact reason for this is unknown, but we speculate that it may be related to the layer complexity of ResNet50. LIME and ProtoShotXAI (ours) have the longest run times whereas ProtoShotXAI is slightly faster than LIME on all except VGG16.

4.5 Few-Shot Exemplary Image Analysis

In a few-shot task, a model is given K exemplary samples from N classes to infer the class of an unknown test sample selected from one of the N classes. In contrast to conventional classification, information is provided to the network during testing (i.e., the K exemplary samples). In the Prototypical few-shot classifier, the K exemplary samples are passed through the neural network and averaged at the feature layer, a concept we have also leveraged for our ProtoShotXAI design. It is well-known that the average of more exemplary samples (i.e., larger K) results in higher classification performance because the average improves the estimate of the class-specific centroid. It follows that single exemplary samples (i.e., 1-shot) are better if they are closer to the class centroid and worse if they are further from it. Therefore, the quality of an exemplary sample in a few-shot task can be quantitatively assessed based on its distance to the class centroid. Unfortunately, why one sample is closer to the class centroid than another remains unanswered.

In this experiment, we demonstrate that ProtoShotXAI is uniquely suited to analyze why one image may be a better or worse exemplary sample for a few-shot task. For each test, we randomly selected a class from ImageNet. We randomly sampled 500 images from

the class then estimate of the class centroid by averaging all 500 images at the feature layer of the network. Next, we find the two nearest and farthest samples from the class centroid. These samples represent the two best exemplary samples and two worst exemplary samples, respectively. ProtoShotXAI is then run on each of the four exemplary images using the 500 images as the support set.

The two best exemplary images, two worst exemplary images, and respective ProtoShotXAI feature attributions are shown Figures 14 and 15 (multiple class best-worst images are provided). We have grouped the figures into two observations. In Figure 14, many of the worst exemplary samples have multiples of the class object in one image. Although this reasoning can be inferred from the images directly, the ProtoShotXAI feature attributions provide detail of which object has positive attributions. ProtoShotXAI also shows that the multiple objects in an image cause significant negative attributions, which moves image's feature representation away from the class center. Alternatively, Figure 15 shows many of the worst exemplary samples have a different perspective or context which cause large negative feature attributions and very little positive attributions. For example, a full image of a person playing an accordion is a better exemplary sample than closeups of the accordion keys. Similarly, the side view of a pencil box is better than the top-view.

In this experiment, results from ProtoShotXAI add useful information to the analysis of exemplary samples. In the case of the multiple objects, ProtoShotXAI identifies the object of interest that is most compelling to the DNN. In the case of perspective, ProtoShotXAI identifies key attributes in the image that contribute to the correct perspective.

5. Discussion

Throughout the experiments section, ProtoShotXAI was demonstrated to be a flexible model exploration tool that produces comparable (and often superior) qualitative and quantitative results. ProtoShotXAI is uniquely straightforward to implement since it does not require training another model (e.g., LIME), or a CNN architecture (e.g., Grad-SHAP and Grad-CAM). ProtoShotXAI only requires a network's feature layer and data from the classes to construct prototypes. Further, comparable feature attribution maps are created using a simple pixel perturbation which could be enhanced with more intelligent perturbation methods.

Although not explicitly presented in this work, we also attempted different variations of ProtoShotXAI. First, we found that the classification weights are needed to produce negative feature attributions because the output of the feature layer often consists of ReLU activations in state-of-the-art neural networks. ReLU activations are either positive or zero, meaning the cosine distance at the feature layer without the classification weights will not produce strong negative feature attributions. In contrast, the Euclidean distance can be used as the similarity score; however, the normalization of cosine distance provided better interpretability across features and was more consistent with the original model response in experiments such as the revolving six. Additionally, the distance between the prototype and query sample can be computed at the output of the final classification layer rather than the weighted feature layer. We found this also produced less favorable results. More specifically, it is common for the final classification layer to have lower dimensionality than the feature layer and a softmax activation function, rather than an unbounded ReLU activation at the

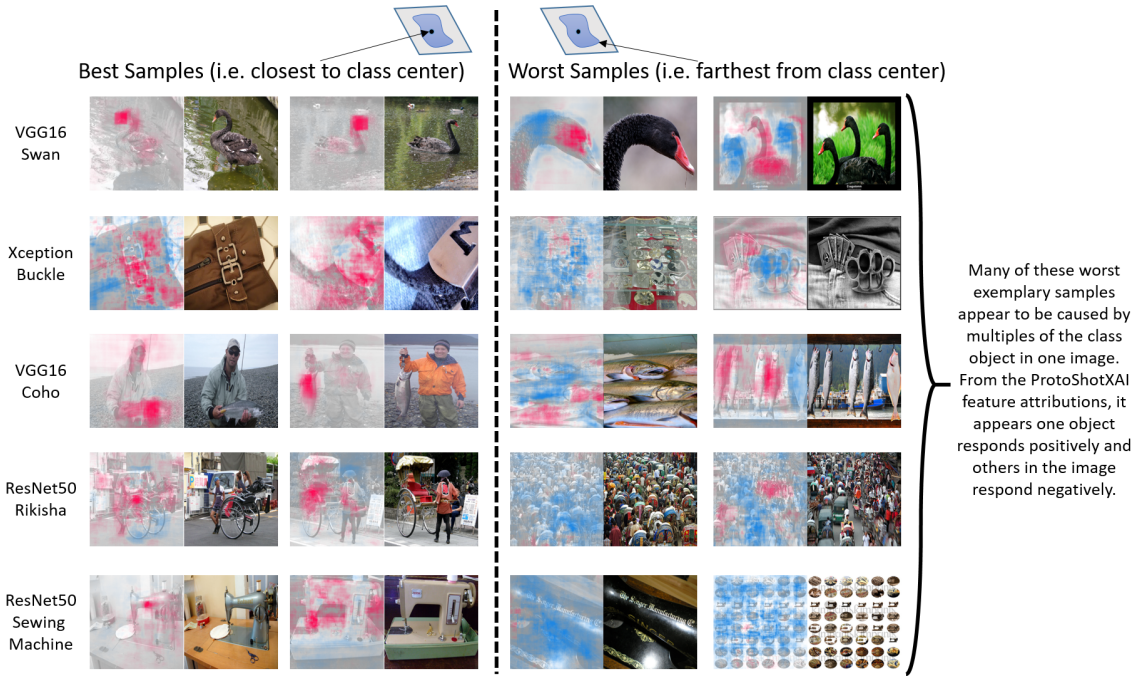


Figure 14: For each class, 500 random images are selected and the two best exemplary samples (left) and two worst exemplary samples (right) are shown with their respective ProtoShotXAI feature attributions. The worst samples have several objects of the class in the image. ProtoShotXAI shows that the classifier focuses on one to respond positively to and responds negatively to many of the surrounding objects.

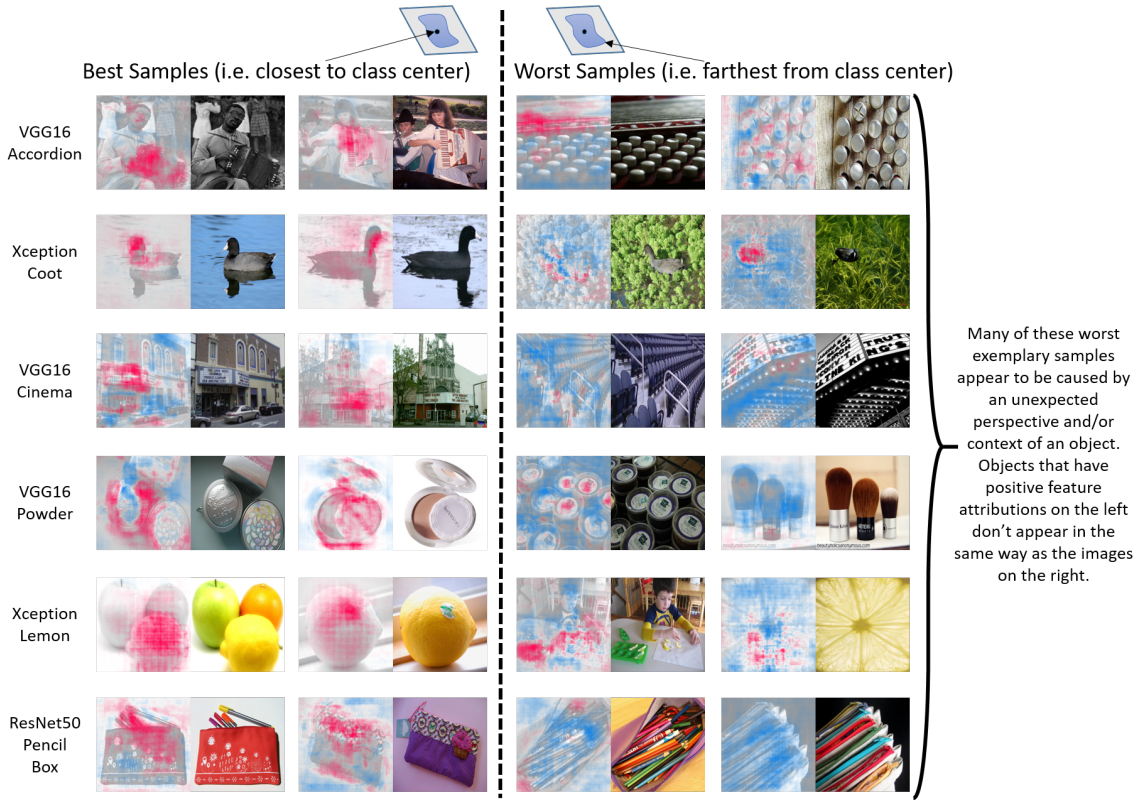


Figure 15: For each class, 500 random images are selected and the two best exemplary samples (left) and two worst exemplary samples (right) are shown with their respective ProtoShotXAI feature attributions. The worst samples have perspective or contextual differences. ProtoShotXAI shows what the classifier focuses on in the the best samples. The large amount of negative attributions in the worst samples show that the DNN is not finding features positively responding to the class.

feature layer. Both the lower dimensionality and bounded activation functions limited the feature attribution interpretability at the classification layer. The class-weighted feature layer produced the best results in our experiments. We also investigated training models to enhance our approach. For example, we implemented an augmentation to the baseline classifier with a few-shot training paradigm. The training objective was to learn adaptive weights between the prototype features and query features such that it would minimize the distance between like classes and maximize the distance between unlike classes. Although the model successfully trained the dynamic weights, it only produced marginally better results. We did not include it here because the benefit of the augmented model was outweighed by the complexity and time cost of having an extra training process. Conveniently, ProtoShotXAI, as presented in this work, does not require any additional training.

6. Conclusions

Machine learning, particularly deep learning, has been shown to provide state-of-the-art performances for numerous benchmarks; however, the explainability of these models and how they reach a decision is a challenging task that limits their utility in some applications. Consequently, it is nearly impossible to quantitatively assess ML approaches' performance to explain black-box models directly. Instead, indirect qualitative and quantitative measures are used for XAI algorithm performance assessment. Qualitative measures can produce misleading results by exploiting an observer's confirmation bias and indirect quantitative measures are imperfect assessments. As a result, there is no one-size-fits-all XAI approach and the best approach depends on the data and application. It was against this background that we proposed the ProtoShotXAI architecture. ProtoShotXAI is an approach to aid ML practitioners in exploring their black-box model and has expanded XAI applicability towards DNN few-shot architectures. We demonstrated on three datasets that ProtShotXAI can create feature attribution maps that are qualitatively comparable, and arguably superior, to state-of-the-art methods. This work also showed how ProtoShotXAI can be extended to higher-level feature analysis by examining the similarity scores as a function of complex changes to the input as demonstrated in the MNIST revolving **six** and Omniglot sub-character removal experiments. We demonstrate both quantitatively and qualitatively that ProtoShotXAI provides more flexibility for feature analysis, model exploration, and applicability to few-shot. Further, the components of the similarity score provide additional insight into model behavior, as demonstrated in the adversarial MNIST experiment. ProtoShotXAI in these specific experiments demonstrates that a user can apply ProtoShotXAI as an exploration tool that will aid in improving the trust in black-box models.

The data used in this work was instrumental in understanding how XAI can be used and evaluated in practice. The digit **eight** was selected due to the convenient presence of other digits within the trace of an **eight**. The image containing the tank and french bulldog was created as a test image without contextual clues and two, and only two, classes. All other data manipulations (i.e., revolving **six**, sub-character analysis in Omniglot, adversarial analysis) were done intelligently. The intelligent manipulations allowed us to make apriori assumptions about a model's outcome that could be validated against the information provided by ProtoShotXAI. Our future work aims to expand the intelligent data manipulations done in this work into more comprehensive datasets to evaluate XAI ap-

proaches further. Specifically, quantitative metrics may be obtained by carefully occluding or removing objects from images in ways that appear visually natural, especially in images that contain multiple objects (classes) of interest. Carefully designed surveys could also help, especially if the surveys account for human-in-the-loop interaction and hypothesis testing. Additionally, ProtoShotXAI is applicable to classification networks for many different data types. Although the scope of this work was limited to images, future work will extend ProtoShotXAI to classification networks with sound, text, and data signals.

Acknowledgements

This work was supported by grants from the Department of Energy #DE-NA0003946, and the National Science Foundation (NSF) CAREER #1943552 Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the sponsors' views. S. Hess was affiliated with the University of Arizona and G. Ditzler was affiliated with the University of Arizona and Rowan University when this work was performed.

Appendix A. Implementation Details

Datasets In this work, we use three commonly used datasets to evaluate the interpretability of our approach: MNIST, Omniglot, and ImageNet. The MNIST dataset is a well-established benchmark for image classification. Although high classification performance can be achieved with very simple models, it is convenient to revisit this dataset in the context of XAI due to the overlap between features of different classes. For example, the number **eight** has a unique cross-over "x"-like feature in the center; however, if the left half of the **eight** were removed then the remaining features appear like a **three**. Similarly, if the diagonal line from the bottom left to the top right of the **eight** were removed then the remaining features appear more like a **five**. We would expect similar conclusions when comparing XAI approaches on sufficiently trained classification networks.

The Omniglot dataset is a character database that is similar to MNIST but is used to benchmark few-shot algorithms. The dataset is composed of 50 different alphabets, and a total of 1,623 character sets (Lake et al., 2011). For each of the 1,623 characters, there are only 20 human-drawn samples which result in a database size of 32,460 samples. The data are originally 105x105 binary images and are resized to 28x28 to reduce input dimensionality. In this dataset, 1,200 characters are used for training/validation and the remaining set of 423 characters are used for a disjoint testing set. Within their respective disjoint sets, images are rotated in multiples of 90 degrees to produce a set of 4,800 classes (96,000 samples) for training/validation and 1,692 classes (423 with 90° rotations) for testing (33,840 samples). To our knowledge, the Omniglot dataset has not been used in any other XAI approach and its application is unique to our work.

ImageNet is a low-resolution image database that is also known as the ILSVRC-12 database (Russakovsky et al., 2015b). Samples vary in size but are often reduced to an image of size 3x224x224 RGB (as in this work) or less. The entire dataset is made up of nearly 14 million samples, but the scale-invariant feature set used in computer vision problems (and in this work) contains 1,000 classes with 1,200 samples per class (i.e., 1.2

million samples total). There is also a subset of the scale-invariant ImageNet, known as Stylized-ImageNet, that we use in this work. Stylized-ImageNet substitutes textures in the images with the goal of creating a training dataset that is biased towards shape instead of texture (Geirhos et al., 2019). For example, the fur from a dog is replaced with textures of skin from an elephant or bricks from a house while maintaining the shape and resemblance of the dog.

Classification Architectures All the XAI approaches evaluated in this work are applied to trained classification architectures, so each dataset was paired with a suitable neural network. For the MNIST dataset, we used the convolutional network shown in Table 6 which has a total of 1.2 million trainable parameters. For the Omniglot dataset, we followed the configuration used in prototypical networks (Snell et al., 2017a) and shown in Table 7 with 112K trainable parameters. Finally, for ImageNet, we used the VGG16, ResNet50, and Xception networks available in the Keras applications library. VGG16 was chosen as a common benchmark between methods because most comparable XAI approaches can be easily applied. VGG16 has 71% classification accuracy and 138 million parameters. Conversely, Xception network has a relatively small number of parameters (22 million) and is one of the best performing networks available in Keras (79% classification accuracy, third only to NASNetLarge with 89M parameters and 83% performance and InceptionResNetV2 with 56M parameters and 80% performance). Due to the complexity and depth of the Xception network (i.e., 126 layers with multiple convolutional layers, separable convolutional layers, and skip layers), only ProtoShotXAI and LIME can be easily applied to this network, and the other methods fail due to limited computational resources. Additionally, ResNet50 was used to compare conventional ImageNet training and a shape-based variant of ImageNet, Stylized ImageNet. The network architectures were equivalent for both ImageNet and Stylized ImageNet, only the weights are different. VGG16 is shown in Table 8. For the specific ResNet50 and Xception network details, we refer the reader to the figures in the original papers (He et al., 2016; Chollet, 2017) or the supplemental material section on Github[1].

Classification Training, Optimization, and Performance To train the MNIST and Omniglot networks, we used the ADAM optimizer (Kingma and Ba, 2015). The learning rate for the MNIST and Omniglot network was initially set to 6×10^{-5} and 1×10^{-3}, respectively. A scheduler was used for the Omniglot experiments to decrease the learning rate by half every 100 epochs for a total of 1,000 epochs. We did not use a scheduler in the MNIST network because a sufficient performance was achieved with a constant learning rate after 100 epochs. For the VGG16, ResNet50, and Xception ImageNet, we used the available pretrained weights with the exception of one experiment where we trained the ResNet50 model from scratch using Stylized-ImageNet data intended to remove texture bias. For the ResNet50 Stylized-ImageNet training, the SGD optimizer was used with momentum. The initial learning rate was set to 0.1 and decreases by 90% every 20 epochs for a total of 40 epochs. For reference, the accuracy of each trained model on their respective dataset are shown in Table 9. Accuracy is reported as the top-1 classification accuracy with the exception of the few-shot network which is reported as the 1-shot, 20-way, classification. As expected, the accuracy on the Stylized-ImageNet dataset is notably lower due to the applied texture randomization.

1. `https://github.com/samuelhess/ProtoShotXAI/`

Model: Convolutional MNIST Model

Layer	Output Shape	# of Parameters
Input	(28, 28, 1)	0
Conv2D (ReLU Activation)	(26, 26, 32)	320
Conv2D (ReLU Activation)	(24, 24, 64)	18496
MaxPooling2D	(12, 12, 64)	0
Dropout 25%	(12, 12, 64)	0
Flatten	(9216)	0
Dense Feature Layer (ReLU Activation)*	(128)	1179776
Dropout 50%	(128)	0
Dense Classification Layer (Softmax Activation)	(10)	1290
Total parameters: 1,199,882		
Trainable parameters: 1,199,882		
Non-trainable parameters: 0		

* The output of this layer corresponds to $\mathbf{f}(\mathbf{x})$ in Section 3.

Table 6: Convolutional MNIST Model

Model: Prototypical Omniglot Model

Layer	Output Shape	# of Parameters
Input	(28, 28, 1)	0
Conv2D (BatchNorm + ReLU Activation)	(28, 28, 64)	896
MaxPooling2D	(14, 14, 64)	0
Conv2D (BatchNorm + ReLU Activation)	(14, 14, 64)	37184
MaxPooling2D	(7, 7, 64)	0
Conv2D (BatchNorm + ReLU Activation)	(7, 7, 64)	37184
MaxPooling2D	(3, 3, 64)	0
Conv2D (BatchNorm + ReLU Activation)	(3, 3, 64)	37184
MaxPooling2D	(1, 1, 64)	0
Flatten (Feature Layer)*	(64)	0
Total params: 112,448		
Trainable params: 111,936		
Non-trainable params: 512		

* The output of this layer corresponds to $\mathbf{f}(\mathbf{x})$ in Section 3.

Table 7: Prototypical Omniglot Model

Model: VGG16 ImageNet Model

Layer	Output Shape	# of Parameters
Input	(244, 244, 3)	0
Conv2D (BatchNorm + ReLU Activation)	(224, 224, 64)	1792
Conv2D (BatchNorm + ReLU Activation)	(224, 224, 64)	36928
MaxPooling2D	(112, 112, 64)	0
Conv2D (BatchNorm + ReLU Activation)	(112, 112, 128)	73856
Conv2D (BatchNorm + ReLU Activation)	(112, 112, 128)	147584
MaxPooling2D	(56, 56, 128)	0
Conv2D (BatchNorm + ReLU Activation)	(56, 56, 256)	295168
Conv2D (BatchNorm + ReLU Activation)	(56, 56, 256)	590080
Conv2D (BatchNorm + ReLU Activation)	(56, 56, 256)	590080
MaxPooling2D	(28, 28, 256)	0
Conv2D (BatchNorm + ReLU Activation)	(28, 28, 512)	1180160
Conv2D (BatchNorm + ReLU Activation)	(28, 28, 512)	2359808
Conv2D (BatchNorm + ReLU Activation)	(28, 28, 512)	2359808
MaxPooling2D	(14, 14, 512)	0
Conv2D (BatchNorm + ReLU Activation)	(14, 14, 512)	2359808
Conv2D (BatchNorm + ReLU Activation)	(14, 14, 512)	2359808
Conv2D (BatchNorm + ReLU Activation)	(14, 14, 512)	2359808
MaxPooling2D	(7, 7, 512)	0
Flatten	(25088)	0
Fully Connected Layer	(4096)	102764544
Fully Connected Feature Layer*	(4096)	16781312
Dense Classification Layer (Softmax Activation)	(1000)	4097000
Total params: 138,357,544		
Trainable params: 138,357,544		
Non-trainable params: 0		

* The output of this layer corresponds to $\mathbf{f}(\mathbf{x})$ in Section 3.

Table 8: VGG16 ImagNet Model

Model	**Dataset**	**Top-1 Classification Accuracy**
Prototypical Convolutional Omniglot*	Omniglot	98.1%
Convolutional MNIST	MNIST	99.2%
VGG16	ImageNet	71.3%
ResNet50	ImageNet	74.9%
Xception	ImageNet	79.0%
ResNet50	Stylized-ImageNet	43.4%

*1-shot, 20-way classification.

Table 9: Test classification accuracy for each model used in the evaluation of ProtoShotXAI

XAI Experimental Protocol The experimental protocol was to train each network with their respective data to achieve sufficient performance. After the network was trained, we constructed the ProtoShotXAI architecture presented in Section 3.1 and demonstrated in Figure 1. We used the feature layer presented in bold for each network in each of the respective tables. When available, we applied the weights of the classification nodes to the features for the respective class. For comparison, each approach is used to explain the same image with the same model. Many approaches display feature attribution weights differently, but we used the approach developed by SHAP for all methods to maintain consistency for comparison. That is, once the feature attribution weights $\mathbf{z}$ are computed, we compute the 99.9 percentile of the absolute values of $\mathbf{z}$. The negative and positive of this value represents the dynamic range of the color axis where red and blue colors indicate positive and negative feature attributions, respectively.

Appendix B. ProtoShotXAI Hyperparameter Study

ProtoshotXAI has three hyperparameter settings for attribution map generation: (1) the number of support samples, (2) the reference value used for pixel perturbation, and (3) the size of the perturbation area. In this section, we independently vary each parameter to demonstrate the effect the parameter has on various images. We have chosen a general set of hyperparameters that produce reasonable results w.r.t. qualitative analysis, which is consistent with several other XAI approaches.

Number of support samples ProtoShotXAI is designed for sample-to-sample comparisons, or comparisons between a query image against an entire class. In the latter case, an entire class is approximated via a prototypical representation of the set of support samples. As observed in the few-shot literature, convergence to the true class prototype is generally achieved by averaging more samples when creating the prototype. Our first hyperparameter experiment demonstrates the ProtoShotXAI attributions w.r.t the number of samples in the support set. Improved stability can be visually observed as the change in attributions from subsequent images decreases. The results are shown in Figure 16. The number of samples varies from 1 to 220 in this experiment and are selected uniformly at random without replacement from the support set. As predicted, the variation in the attribution result is larger when the number of samples is lower. It is observed that the difference between the attribution images becomes visually negligible as the number of samples approaches 60. This observation is consistent across several different images we investigated (see Figure 16). Figure 17 shows the variance of the feature attributions for each network architecture as the number of exemplary samples increases. The results are averaged over 5 images and we observe that variance quickly decreases from 1-20 shot and is nearly zero after 100 shot. We used 100 samples in Section 4's experiments as a hyperparameter in our work, ensuring consistent results w.r.t a class support set.

Reference Perturbation Value Our second hyperparameter experiment examines the value set for the perturbation pixel in Algorithm 1. There are several approaches that could be used to set the perturbed pixel value. Such approaches could include setting the perturbation to a zero reference (i.e., black color), maximum reference (i.e., white color), average value (i.e., brownish color), any value on $[0, 1]$, or even a complex perturbation that

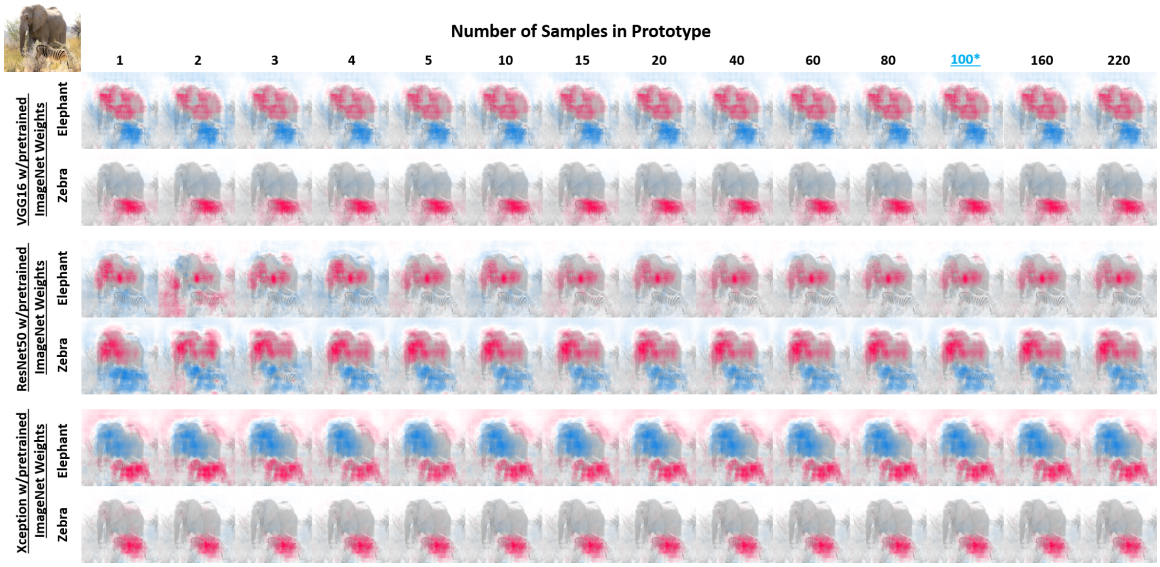


Figure 16: ProtoShotXAI feature attributions for an African elephant and zebra using VGG16, ResNet50, and Xception network classifiers. From left to right, the number of samples included in the reference support set is varied from 1 to 220. More variance is seen in the lower number of samples and all the attributions appear to converge by 60 sample support set. 100 sample support set was used throughout our work.

is a function of the pixel value. Figure 18 shows the ProtoShotXAI feature attributions using zero and an average value references. The attributions are very similar between the two methods; however, the zero reference method tends to have larger attribution weights across the image. The larger weights are most visible in the background attributions on the tank/bulldog image. We suspect this behavior of the zero reference is due to the fact that the perturbation value is at the extreme of pixel values. Thus, the result will likely cause more variation in the feature manifold of the network.

Size of Perturbation Area Our third hyperparameter experiment varies the size of the perturbation area. The perturbation area is defined by a square pixel region centered around a pixel of interest. The entire area is replaced by the reference pixel to influence larger deviations in the feature manifold w.r.t. a spatial area of the image. Each pixel in the image is evaluated as a pixel of interest so the perturbation is equivalent to a moving filter and has a spatial averaging-like effect. The pad represents the number of pixels that surrounds the center pixel of interest in both vertical and horizontal spatial dimensions. Figure 19 shows the results of increasing the pad size between 1 and 50 for the four networks trained on ImageNet. The support set was set to 100 samples and the average value was used for perturbation. As the pad size increases, the ProtoShotXAI attributions cover more of the object under evaluation (i.e., the elephant or zerbra); however, the detail of the attribution is reduced. For networks based around the ImageNet classification of 224×224 RGB images we found that a pixel pad of 15 works qualitatively well.

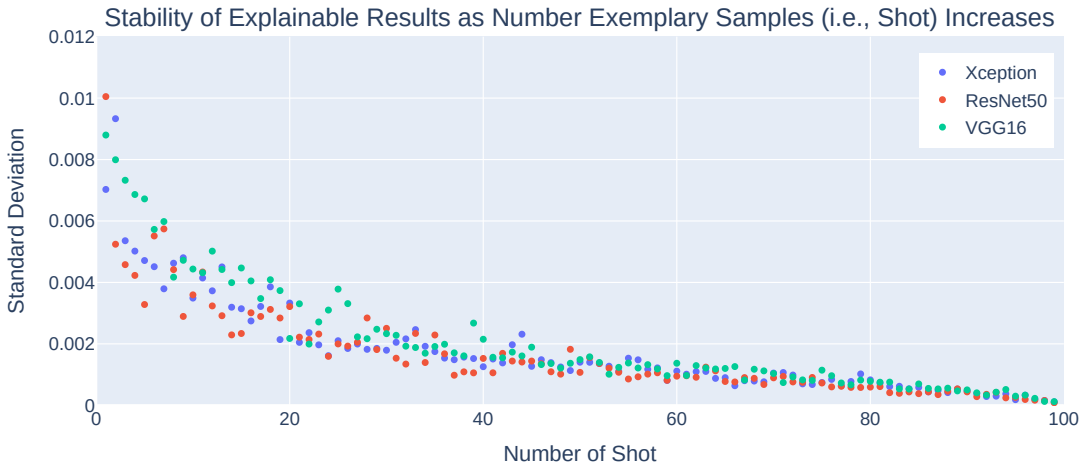


Figure 17: Standard deviation of ProtoShotXAI feature attributions as a function of the number of exemplary samples (i.e., shot). The results correspond to the feature attribution average of 5 images passed through each classifier. As the number of exemplary samples increase, the results of the ProtoShotXAI feature attributions become more stable. A standard deviation of zero indicates that there is no change to the feature attributions with a different set of randomly selected exemplary samples. The standard deviation for both networks quickly decreases from 1-20 shot and is nearly zero after 100 shot.

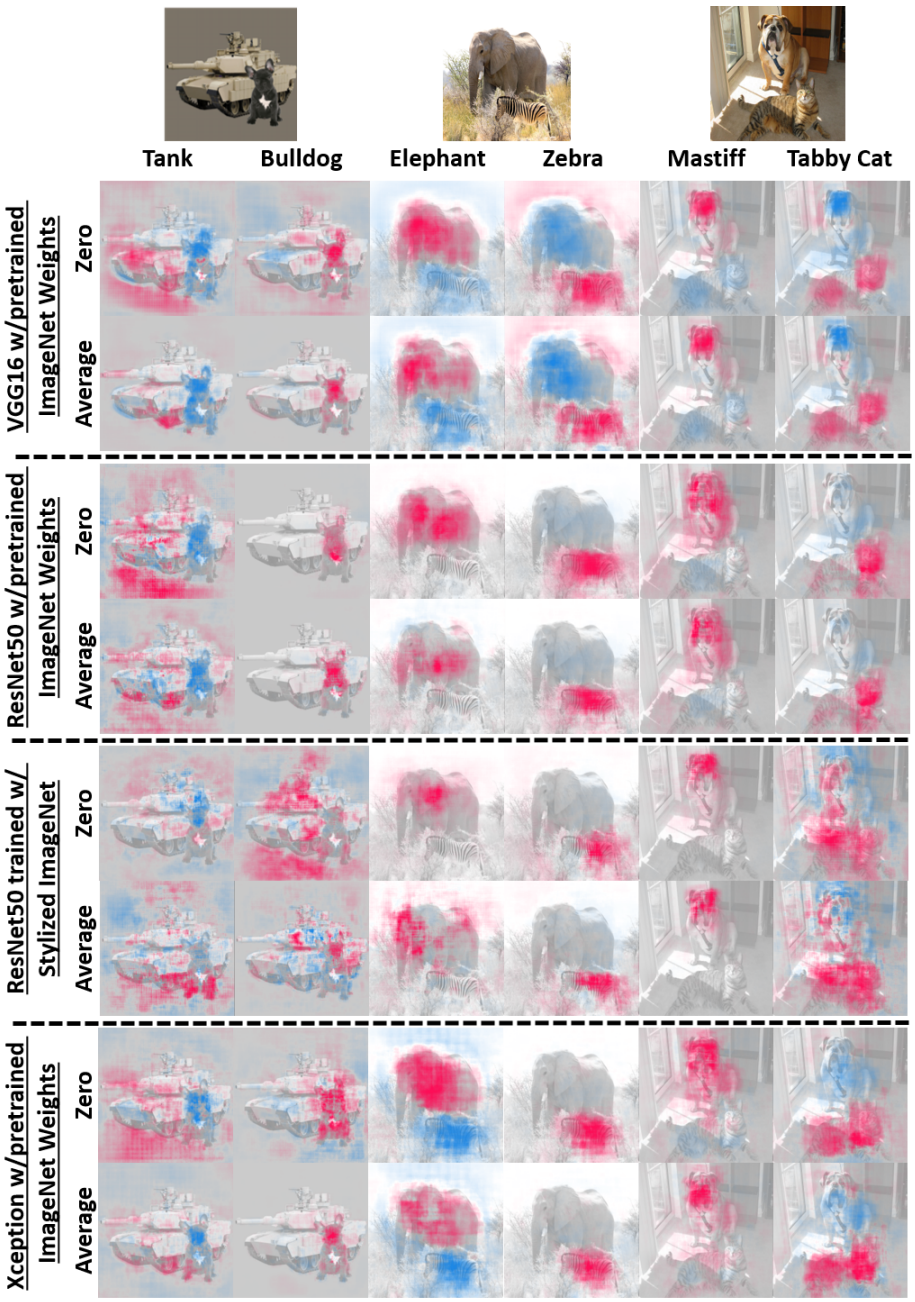


Figure 18: ProtoShotXAI feature attributions for three images using VGG16, ResNet50, and Xception network classifiers. For each image and each network, both the zero reference and average value reference for pixel perturbation are shown. Results are visually similar between the two approaches but the zero reference method has more attribution weight magnitude to the background in the tank/bulldog image.

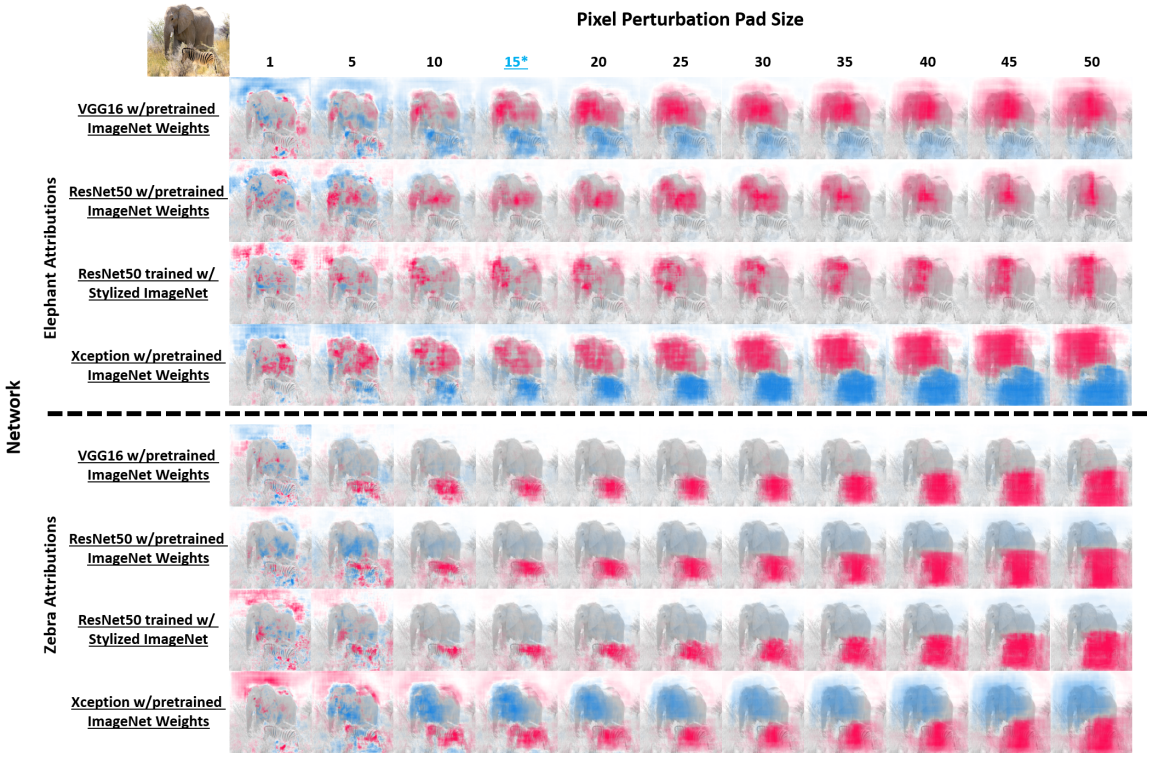


Figure 19: Comparison of ProtoShotXAI feature attributions w.r.t different perturbation areas for various networks trained for ImageNet classification. The top half of the figure demonstrates attributions w.r.t. an African elephant classification and the bottom half is w.r.t. zebra classification. There is a trade off between the pad size. As the pad size increases, the attributions cover more of the object under evaluation but the detail of the attribution is reduced. Unless specified otherwise, the pixel pad of 15 was used throughout our work.

References

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4845–4854, 2019.

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *International Conference on Learning Representations*, 2018.

Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011.

Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, pages 1–60, 2023.

Steven Bramhall, Hayley Horn, Michael Tieu, and Nibhrat Lohia. Qlime-a quadratic local interpretable model-agnostic explanation approach. *SMU Data Science Review*, 3(1):4, 2020.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

Chao Chen, Yifan Shen, Guixiang Ma, Xiangnan Kong, Srinivas Rangarajan, Xi Zhang, and Sihong Xie. Self-learn to explain siamese networks robustly. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1018–1023. IEEE, 2021.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *International Conference on Learning Representations*, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

Radwa ElShawi, Youssef Sherif, Mouaz Al-Mallah, and Sherif Sakr. Ilime: Local and global interpretable model-agnostic explainer of black-box decision. In *European Conference on Advances in Databases and Information Systems*, pages 53–68. Springer, 2019.

Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott M Lundberg, and Su-In Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature Machine Intelligence*, pages 1–12, 2021.

Andrea Fedele, Riccardo Guidotti, and Dino Pedreschi. Explaining siamese networks in few-shot learning for audio data. In Poncelet Pascal and Dino Ienco, editors, *Discovery Science*, pages 509–524. Springer Nature Switzerland, 2022.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Michael Fink. Object classification from a single example utilizing class relevance metrics. *Advances in neural information processing systems*, 17:449–456, 2004.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bygh9j09KX`.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2014.

Riccardo Guidotti. Evaluating local explanation methods on ground truth. *Artificial Intelligence*, 291:103428, 2021.

Karthik S Gurumoorthy, Amit Dhurandhar, Guillermo Cecchi, and Charu Aggarwal. Efficient data representation by selecting prototypes with importance weights. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 260–269. IEEE, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE CVPR*, pages 770–778, 2016.

Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani Srivastava. How can i explain this to you? an empirical study of deep neural network explanation methods. *Advances in Neural Information Processing Systems*, 2020.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learn. Wrksp*, 2015.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

Huayu Li and Gregory Ditzler. Targeted Data Poisoning Attacks Against Continual Learning Neural Networks. In *IEEE/INNS International Joint Conference on Neural Networks*, 2022.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

Erwan Le Merrer and Gilles Trédan. Remote explainability faces the bouncer problem. *Nature Machine Intelligence*, 2:529–539, 2020.

Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

Tomi Peltola. Local interpretable model-agnostic explanations of bayesian predictive models via kullback-leibler projections. *2nd Workshop on Explainable Artificial Intelligence*, 2018.

Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

Han Qiu, Yi Zeng, Qinkai Zheng, Shangwei Guo, Tianwei Zhang, and Hewu Li. An Efficient Preprocessing-based Approach to Mitigate Advanced Adversarial Attacks. *IEEE Transactions on Computers*, 2021.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

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

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015a. doi: 10.1007/s11263-015-0816-y.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015b.

Koosha Sadeghi, Ayan Banerjee, and Sandeep Gupta. A System-Driven Taxonomy of Attacks and Defenses in Adversarial Machine Learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(4):450–467, 2020.

Tyler Scott, Karl Ridgeway, and Michael C Mozer. Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Sharath M Shankaranarayana and Davor Runje. Alime: Autoencoder based approach for local interpretability. In *International conference on intelligent data engineering and automated learning*, pages 454–463. Springer, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 2017a.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 2017b.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6021–6029, Apr. 2020.

Florian Tramér, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The Space of Transferable Adversarial Examples. *arXiv:1704.03453*, 2017.

Lev Utkin, Maxim Kovalev, and Ernest Kasimov. Explanation of siamese neural networks for weakly supervised learning. *Computing and Informatics*, 39(6):1172–1202, 2020.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

Tom Vermeire, Dieter Brughmans, Sofie Goethals, Raphael Mazzine Barbossa de Oliveira, and David Martens. Explainable image classification with evidence counterfactual. *Pattern Analysis and Applications*, pages 1–21, 2022.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, and others. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7278–7286, 2018.

Meng Ye and Yuhong Guo. Deep triplet ranking networks for one-shot recognition. *arXiv preprint arXiv:1804.07275*, 2018.

Xiaomeng Ye, David Leake, William Huibregtse, and Mehmet Dalkilic. Applying class-to-class siamese networks to explain classifications with supportive and contrastive cases. In *International conference on case-based reasoning*, pages 245–260. Springer, 2020.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

Muhammad Rehman Zafar and Naimul Mefraz Khan. Dlime: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems. In *In proceeding of ACM SIGKDD Workshop on Explainable AI/ML (XAI) for Accountability, Fairness, and Transparency*. ACM, 2019.

Yichi Zhang, Bryan Pardo, and Zhiyao Duan. Siamese style convolutional neural networks for sound search by vocal imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(2):429–441, 2018.