Independent Prototype Propagation Graph for Compositional Zero-Shot Recognition

Frank Ruis

Supervisors: Dr. D. Bucur
Dr. ir. G.J. Burghouts
Dr. Hanyuan Hang

MSc Computer Science, Data Science & Technology
University of Twente

August 2021

Preface

This thesis is the result of an internship at TNO, where I worked on compositionality and knowledge graphs in zero-shot learning. We started out with the goal to develop a method that tackles the problem known as Compositional Zero-Shot Learning (CZLS). In CZSL, the goal is to compose a set of visual primitives, such as the color (attribute) or shape (object) of a shoe, into a set of novel compositional classes that the model has not seen before. Due to the complexity of the problem, CZSL has been constrained to a single attribute and object label (e.g. just one color and one global shape) per image. Progress on the problem was quicker than anticipated, leading to a paper that is currently in review at the Conference on Neural Information Processing Systems (NeurIPS), which starts on page 1 of this thesis. The paper was a sort of natural conclusion to the aforementioned problem, but my internship at TNO had not yet run its course. Therefore, I spent the remainder of my internship extending the method to an unconstrained version of the problem, the result of which can be found from page 15 onwards.

I would like to thank my supervisors, Gertjan and Doina, for their valueable feedback and help with writing the NeurIPS paper, and Hanyuan, who was willing to join my graduation committee and provide me with lightning-fast feedback on my thesis when I was struggling to plan my graduation at the last minute. I thoroughly enjoyed my time at TNO, so much so that I will be joining them again after my graduation as a computer vision researcher.

Frank Ruis
August 2021

Abstract

Humans are good at compositional zero-shot reasoning; someone who has never seen a zebra before could nevertheless recognize one when we tell them it looks like a horse with black and white stripes. Machine learning systems, on the other hand, usually leverage spurious correlations in the training data, and while such correlations can help recognize objects in context, they hurt generalization. To be able to deal with underspecified datasets while still leveraging contextual clues during classification, we propose ProtoProp, a novel prototype propagation graph method. First we learn prototypical representations of objects (e.g., zebra) that are conditionally independent w.r.t. their attribute labels (e.g., stripes) and vice versa. Next we propagate the independent prototypes through a compositional graph, to learn compositional prototypes of novel attribute-object combinations that reflect the dependencies of the target distribution. The method does not rely on any external data, such as class hierarchy graphs or pretrained word embeddings. We evaluate our approach on AO-Clevr, a synthetic and strongly visual dataset with clean labels, and UT-Zappos, a noisy real-world dataset of fine-grained shoe types. We show that in the generalized compositional zero-shot setting we outperform state-of-the-art results, and through ablations we show the importance of each part of the method and their contribution to the final results.

In the final section, we extend our method to a more challenging multi-attribute setting, where images may contain any arbitrary number of attributes, instead of just one label per image. With a small change to the loss function, and some simplifications to the attribute propagation step, we reach generalized zero-shot classification results on par with the state of the art on a fine-grained bird dataset and a course-grained animal dataset. We find that the commonly used zero-shot benchmarks for this setting may have reached a performance ceiling.

Table of contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Related Work . . . 2

2 Method **5**
- 2.1 Prototype-based representations . . . 5
- 2.2 Attribute-object independence . . . 7
- 2.3 Prototype propagation graph . . . 8

3 Experiments **10**
- 3.1 Implementation . . . 10
- 3.2 Evaluation . . . 10
- 3.3 Results . . . 11
- 3.4 Ablations . . . 12
- 3.5 Limitations . . . 13

4 Multi-attribute extension **15**
- 4.1 Method . . . 15
- 4.2 Experiments . . . 18
  - 4.2.1 Evaluation . . . 18
  - 4.2.2 Results . . . 19
- 4.3 Limitations . . . 20
- 4.4 Conclusion . . . 21

References **22**

Appendix A **26**
- A.1 Hyperparameters . . . 26
- A.2 Hilbert-Schmidt Independence Criterion . . . 26
- A.3 Dataset Issues . . . 27
  - A.3.1 AO-Clevr . . . 27
  - A.3.2 UT-Zappos . . . 27
- A.4 Error Analysis . . . 27
  - A.4.1 AO-Clevr . . . 27
  - A.4.2 UT-Zappos . . . 29

A.4.3 CUB 30
A.4.4 AWA2 32
A.5 AO-Clevr Results 33

Chapter 1

Introduction

1.1 Motivation

As humans, hearing the phrase 'a tiny pink penguin reading a book' can conjure up a vivid image, even though we have likely never seen such a creature before. This is because humans can compose their knowledge of a small number of visual primitives to recognize novel concepts (Hebart et al., 2020), a property which Lake et al. (2017) argue is one of the key building blocks for human intelligence missing in current artificial intelligence systems. Machines, on the other hand, are largely data-driven, and usually require many labeled examples from various viewpoints and lighting conditions in order to recognize novel concepts. Since visual concepts follow a long-tailed distribution (Liu et al., 2019b; Salakhutdinov et al., 2011), such an approach makes it near impossible to gather sufficient examples for all possible concepts. Compounded by that, in the absence of sufficient data, vanilla convolutional neural networks will use any correlation they can find to classify training samples, even when they are spurious (Kim et al., 2019). In this work, we aim to tackle both of these issues.

Compositional Zero-Shot Learning (CZSL) (Misra et al., 2017) is the problem of learning to model novel objects and their attributes as a composition of visual primitives. Previous works in CZSL (Li et al., 2020; Misra et al., 2017; Purushwalkam et al., 2019) largely ignore the dependencies between classes with shared visual primitives, and the spurious correlations between attributes and objects. More recently, Atzmon et al. (2020) tackle the latter by ensuring conditional independence between attribute and object representations, while Naeem et al. (2021) explicitly promote dependencies between the primitives and their compositions. While the independence approach improves generalization, it hurts accuracy on seen classes by removing useful correlations. The explicit dependencies, on the other hand, can share these useful correlations with unseen classes, but there will always be some that are spurious, hurting generalization.

In this work, we propose to take advantage of the strengths of both approaches, respectively, by learning independent visual representations of objects and attributes, and by learning their compositions for the target classes. First, we represent visual primitives by learning local independent prototypical representations. Prototype networks (Snell et al., 2017) learn an embedding function, where inputs of the same class cluster around one prototypical represen-

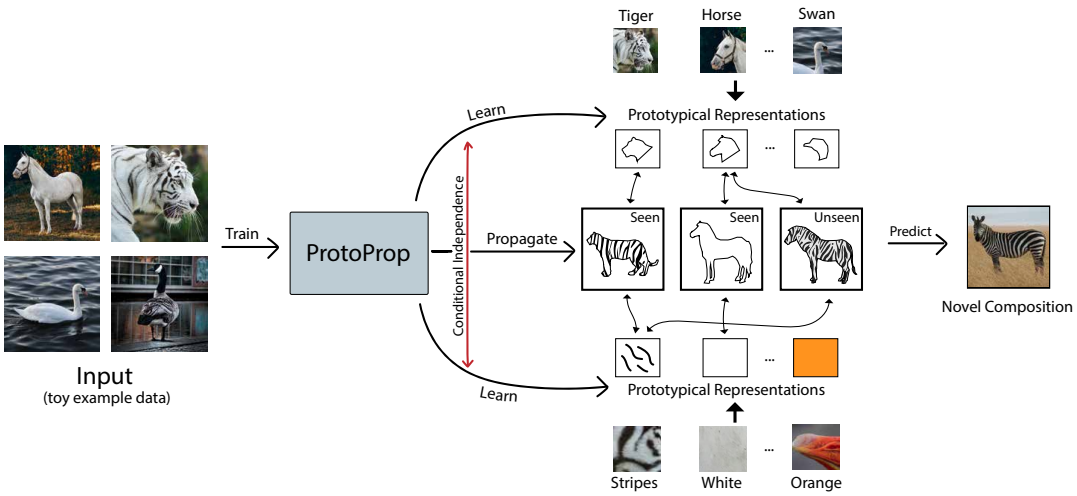

Fig. 1.1 **ProtoProp** A sketch of our proposed method; we learn conditionally independent prototypical representations of visual primitives in the form of objects (e.g., horse) and attributes (e.g., stripes). The prototypes are then propagated through a compositional graph, where they are combined into novel compositional prototypes to recognize both seen and unseen classes (e.g., zebra).

tation of that class. Here we adopt such a function for learning prototypes of objects and attributes. Next, we leverage a compositional graph to learn the dependencies between the independent prototypes on the one hand, and the desired seen and unseen classes on the other, by propagating the prototypes to compositional nodes. Here, the compositional graph allows some information to be shared between objects that share attributes, e.g., between tigers and zebras. The proposed method, ProtoProp, is outlined in Figure 1.

Our main contributions are: 1) We propose a novel graph propagation method that learns to combine local, independent, attribute and object prototypes into one compositional prototype that can accurately detect unseen compositional classes. 2) A spatial attention-based pooling method that allows us to obtain differentiable attribute and object patches for use in an independence loss function. 3) Our method effectively deals with bias from an underspecified dataset by learning conditionally independent representations that then take on the dependencies of the desired target distribution. 4) We validate through ablations the importance of each part of the method (local prototypes vs semantic embeddings, independence loss, backbone finetuning) and their contribution to the final results. 5) We show that we improve on state-of-the-art results on two challenging compositional zero-shot learning benchmarks: 2.5 to 20.2% harmonic mean improvement on AO-Clevr (Atzmon et al., 2020) and 3.1% harmonic mean improvement on UT-Zappos (Yu and Grauman, 2014) compared to the best existing method.

1.2 Related Work

Compositional zero-shot learning (CZSL) methods aim to recognize unseen compositions from known visual primitives in the form of attributes and objects. One line of work considers

embedding the visual primitives in the image feature space. Misra et al. (2017) use the weight vectors of linear SVMs as embeddings for the visual primitives, which they transform to recognize unseen compositions. Li et al. (2020) look at attribute-object compositions through the lens of symmetry inspired by group theory. A different line of work considers a joint embedding function on the image, attribute, and object triplet, allowing the model to learn dependencies between the image and its visual primitives. Purushwalkam et al. (2019) train a set of modular networks together with a gating network that can 'rewire' the classifier conditioned on the input attribute and object pair. Atzmon et al. (2020) take a causal view of CZSL, trying to answer which intervention caused the image. They apply conditional independence constraints to the representations of the visual primitives, sacrificing accuracy on the seen data but performing well on unseen data by removing correlations that are useful but hurt generalization to novel compositions. Naeem et al. (2021), on the other hand, explicitly promote the dependency between all primitives and their compositions within a graph structure, though they rely on pretrained semantic embeddings which may differ in distribution from the visual concepts they describe.

Our proposed method combines the strengths from Atzmon et al. (2020) and Naeem et al. (2021) by first learning conditionally independent representations of the visual primitives, which are then propagated through a compositional graph to learn the dependency structure of the desired target distribution. Unlike most other methods, we are not reliant on pretrained word embeddings, but instead learn the representations for our visual primitives directly from the training data. Instead of a linear kernel like Atzmon et al. (2020), we use a Gaussian kernel for our independence loss as we find that its ability to capture higher-order statistics is beneficial in terms of accuracy. Like Naeem et al. (2021), we train our model fully end-to-end, including the feature extractor, as learning a good embedding is often more beneficial than an overly complicated method applied to suboptimal embeddings (Tian et al., 2020).

Prototypical networks (Snell et al., 2017) aim to learn an embedding function where inputs of the same class cluster around one prototypical representation of that class. They measure L2 distance between samples in pixel space, which is sensitive to non-semantic similarities between images, such as objects from different classes with a similar background and light conditions. Li et al. (2018a) move the similarity metric to latent space and utilise an autoencoder to directly visualize the prototypes, but they only test on simple MNIST-like benchmarks. Chen et al. (2019) extend the method to multiple local prototypes per class, where prototypes are compared to local image patches instead of the entire average-pooled output of a CNN, and evaluate on the more complicated fine-grained bird classification dataset CUB (Wah et al., 2011). We take a similar local-prototype approach, but we use direct attribute and object supervision, enabling parameter sharing between classes and allowing the prototypes to be used for zero-shot classification. While for most methods these prototypes are used for the final classification, in our case they are an intermediate representation.

Graph neural networks (GNNs) are models that can work directly on the structure of a graph. They have been first proposed by Gori et al. (2005), later elaborated upon by Scarselli et al. (2008) and popularised by Kipf and Welling (2017) in their work on the Graph Convolutional Neural Network (GCN). GNNs grow more popular every year, and a lot of improvements have been proposed recently (Wu et al., 2020). Just like the GCNs were inspired by CNNs, most improvements are inspired by other areas of deep learning such as attention mechanisms in

Graph Attention Networks (Veličković et al., 2018), but this jump from existing deep learning fields to GNNs has overlooked simpler methods. Wu et al. (2019) have taken a step back and stripped down GNNs to their simplest parts by removing nonlinearities and collapsing weight matrices until all that was left was feature propagation followed by a linear model. In the same vein, Huang et al. (2021) show that a simple Multilayer Perceptron (MLP) ignoring graph structure followed by a label propagation post-processing step often greatly outperforms GNNs. These simplified methods are mostly limited to transductive settings (test nodes are available at train time) and graphs that exhibit strong homophily (similar nodes are connected). Many real-world graphs fit those criteria, including the graph we use in this work.

Chapter 2

Method

In compositional zero-shot learning (CZSL), we have a set of images X and compositional classes with compositional labels $Y \subseteq A \times O$, where A is a set of attribute labels and O a set of object labels. The labels are subdivided into $Y = Y_s \cup Y_u$, where Y_s are the seen labels for the training set and Y_u unseen labels for the validation and test sets, with $Y_s \cap Y_u = \emptyset$. We denote the training data consisting of only seen compositional classes as X_s. Finally, CZSL assumes that each attribute and object is seen in at least one training data point, or more formally $\forall p \in A \cup O \; \exists y \in Y_s$ s.t. $p \cap y \neq \emptyset$.

2.1 Prototype-based representations

In this section we describe how we learn local attribute and object prototypes, before they are propagated and combined into compositional prototypes in Section 2.3. Prototypes are an average representation of a target class, with strong generalization and interpretability properties. We employ a local prototype approach similar to Chen et al. (2019), though we use direct supervision, encouraging the feature extractor to learn local image representations that encode the attribute and object labels. We use a ResNet-18 (He et al., 2016) backbone for fair comparison to other CZSL methods, but the method is backbone agnostic.

Figure 2.1 shows an example of a prototype layer. The layer has a set of prototype vectors $\mathbf{P} = \{p_j\}_{j=1}^{k}$, $p_j \in \mathbb{R}^C$ where each target class is represented by one prototype. $k = |A|$ or $k = |O|$ is the number of attribute or object targets respectively. The layer takes as input the $H \times W \times C$ output of a CNN just before its final pooling layer, and calculates a compatibility score $s = \max_{x_{ij} \in \mathbb{R}^C} \langle x_{ij}, p_k \rangle$ (e.g., cosine similarity, L2 norm, dot product) between the input patches x_{ij} over the spatial dimensions $H \times W$ and the prototypes. We find that a dot product similarity metric leads to better generalization. To save on parameters and avoid overfitting, we do not use a fully connected layer. Instead, the maximum patch-prototype similarity score is used directly as the compatibility score for the corresponding prototype's class. This compatibility score is optimized through a cross-entropy loss function:

$$\mathcal{L}_{CE} = \frac{1}{|S|} \sum_{s,y \in S} -\log \left(\frac{\exp(s[y_i])}{\sum_j^{|s|} \exp(s[j])} \right) \tag{2.1}$$

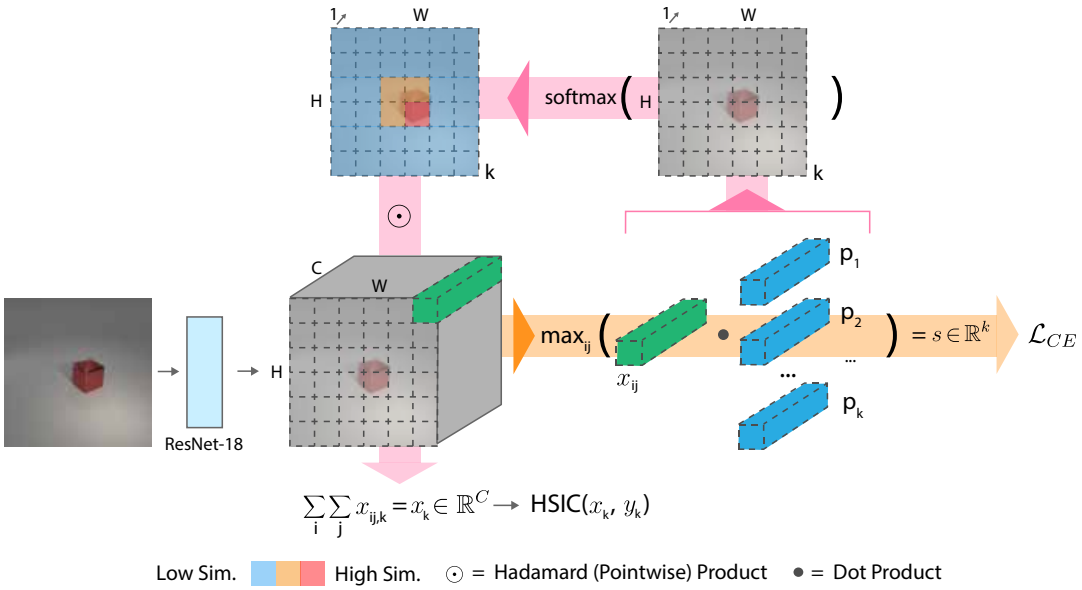

Fig. 2.1 **Local prototypes with softmax pooling**: Local image patches x_{ij} from the layer just before the average pooling layer of a ResNet-18 are compared to prototype vectors p_k through a similarity function, here a dot product. This outputs a compatibility score s, which is optimized via $\mathcal{L}_{CE}$ (cf. Section 2.1). Similar to spatial attention, these patch-prototype compatibility scores are passed through a softmax function and via Hadamard (pointwise) product a weighted sum z_k of the original feature map is calculated, which are passed to the HSIC function to promote independence between object and attribute prototypes (cf. Section 2.2)

Here, with f_p as our prototype layer, $S = \{s, y \in (f_p(X_s) \mid Y_s)\}$ is the set of prototype similarity scores and target labels for the training data. $i \in \{0, 1\}$ indicates if we are training on attribute or object labels respectively.

Cross-entropy loss naturally encourages clustering between samples of the same class and separation from other classes, but we find that an additional loss to push these properties even more is beneficial. For that purpose we adopt the cluster and separation costs proposed by Chen et al. (2019):

$$\text{Clst} = \frac{1}{|X|}\sum_{i=1}^{|X|} \min_{j:p_j \in \mathbf{P}_{y_i}} \min_{z \in x_{ij}} \|z - p_j\|_2^2, \qquad \text{Sep} = -\frac{1}{|X|}\sum_{i=1}^{|X|} \min_{j:p_j \notin \mathbf{P}_{y_i}} \min_{z \in x_{ij}} \|z - p_j\|_2^2. \quad (2.2)$$

The costs Clst and Sep encourage a higher degree of clustering between prototypes of the same class and a greater distance to prototypes of different classes, respectively. The separation cost is only applied to the object prototypes, as the attribute prototypes benefit from less distant representations.

Our Hilbert-Schmidt Independence Criterion (HSIC) loss (details in Section 2.2) requires the input patches with the highest prototype similarity scores as its input, but that would require the use of the argmax function which has a gradient of 0 almost everywhere. Similar to spatial attention, we use the softmax function over the similarity map as a differentiable proxy. The softmax outputs used as weights in a weighted sum z_k of the input patches x_{ij} serve as an approximation of the input patch with the highest similarity score (cf. Figure 2.1), and ensure that each patch containing information about attribute or object labels is affected by the independence loss proportional to the strength of their appearance (i.e. their similarity score).

2.2 Attribute-object independence

To combat the bias in the training data we leverage a differentiable independence metric such that the model can learn representations for the attributes of images that are uniformly distributed w.r.t. their object labels and vice versa.

The Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2007) is a kernel statistical test of independence between two random variables A and B. In the infinite sample limit $\text{HSIC}(A,\ B) = 0 \iff A \perp\!\!\!\perp B$ as long as the chosen kernel is universal in the sense of Steinwart (2001). We use a Gaussian kernel, as we find that its ability to capture higher-order statistics is beneficial in terms of accuracy, and follow Gretton et al. (2007) in setting the kernel size to the median distance between points. See Appendix A.2 for more details. We define our independence loss function as follows, slightly deviating from Atzmon et al. (2020):

$$\mathcal{L}_{hsic} = \lambda_h \frac{\text{HSIC}(z_a,\ O) + \text{HSIC}(z_o,\ A)}{2} \quad (2.3)$$

where λ_h is a hyperparameter controlling the contribution of $\mathcal{L}_{hsic}$ to the final loss, z_a is the softmax-pooled output of the attribute prototype layer (cf. Section 2.1), z_o the softmax-pooled

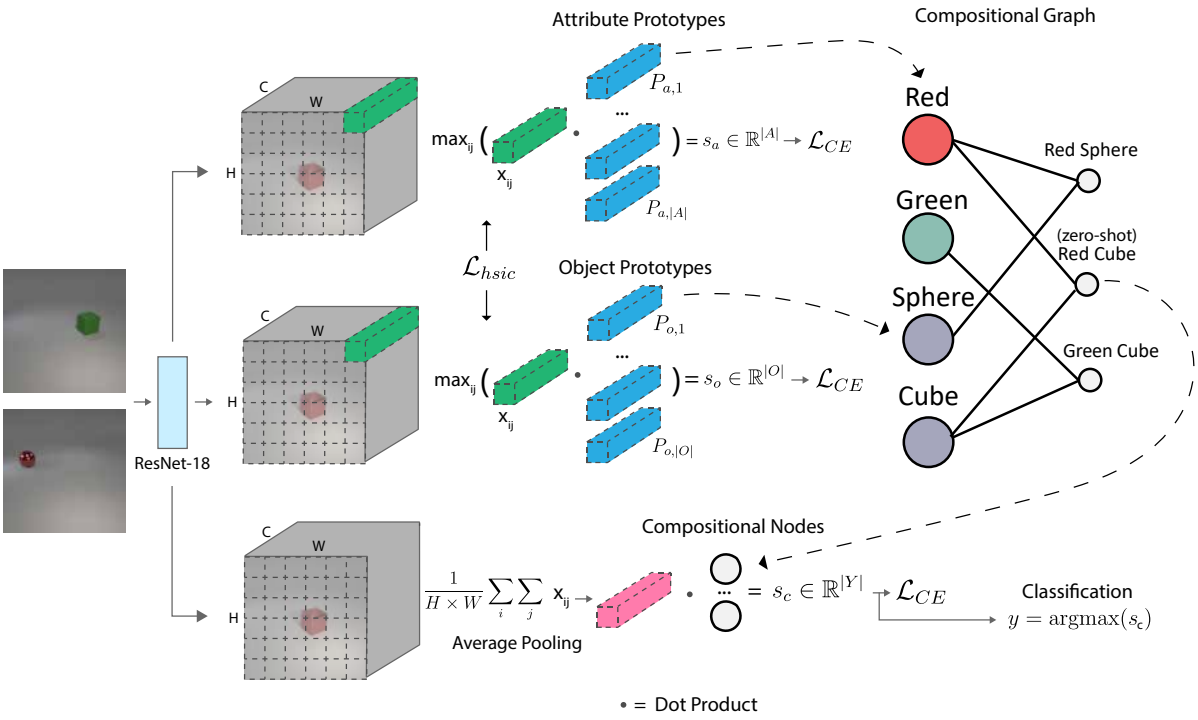

Fig. 2.2 **ProtoProp: overview of proposed method.** Local attribute and object prototypes are trained with independence (HSIC) loss and mapped to nodes in a simplified graph neural network, which learns to combine them into compositional prototypes of both seen and unseen classes. The maximum score in the dot product similarity map between the compositional prototypes and average-pooled backbone output is the final classification prediction.

output of the object prototype layer, and O and A the corresponding one-hot encodings of the object and attribute labels respectively.

2.3 Prototype propagation graph

Figure 2.2 shows the full architecture. Two types of prototype layers are trained, one on the object labels and one on the attribute labels, leaving us with centroids with confounding information removed. The prior knowledge about the compositional target classes and the attributes they afford can be represented as a compositional graph $G = (\mathbf{P}_a \cup \mathbf{P}_o \cup \mathbf{C}_y, E)$. It consists of the attribute prototypes $\mathbf{P}_a$, object prototypes $\mathbf{P}_o$, and compositional classes $\mathbf{C}_y \subseteq A \times O$ (a subset of which have no training samples). The edges are undirected, $E = \{x, y \mid x \in \mathbf{P}_a \cup \mathbf{P}_o,\ y \in \mathbf{C}_y : x \in y\}$, i.e. all attribute (e.g., red) and object (e.g., cube) nodes are connected to the compositional classes they are a part of (e.g., red cube). This corresponds to a bipartite graph with the attribute and object prototypes in one set and the compositional classes in the other. In the case of AO-Clevr it is a complete bipartite graph, but for real-world datasets such as UT-Zappos this is not the case as, e.g., not all shoes are available in all materials.

As defined above, the conditionally independent prototypes are mapped directly to the nodes of the graph (cf. Figure 2.2), and through shared weights a Graph Neural Network (GNN) learns to propagate the prototypes to the compositional nodes to form new compositional prototypes. The compositional nodes are initialized with zeros. Our GNN is a 2-layer GCN (Kipf and Welling, 2017); inspired by SGC (Wu et al., 2019) we remove all nonlinearities as we find that simple linear combinations lead to better generalization:

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta} \tag{2.4}$$

Here $\mathbf{X}$ are the input nodes, $\mathbf{\Theta}$ a learnable weight matrix, trained jointly with the rest of the architecture, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacency matrix with inserted self-loops, and $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ its diagonal degree matrix.

We then compute another dot product similarity map s_c between the compositional prototypes and the average-pooled output of the backbone, which is again optimized via cross-entropy loss function. Our final compositional classification is $y = \text{argmax}(s_c)$, i.e., the class corresponding to the compositional prototype with the maximum similarity score. The shared weights in the GNN ensure that the model learns a general composition of attributes and objects that can generalize to unseen compositional classes. By initializing the node features with independent prototypes, the graph is able to learn the dependencies encoded by the compositional graph, including the novel zero-shot classes, instead of the biases of the training data.

Chapter 3

Experiments

3.1 Implementation

We implement our model using the PyTorch library (Paszke et al., 2019), using some of the boilerplate code provided by Nagarajan and Grauman (2018) and Purushwalkam et al. (2019). The GNN is implemented using PyTorch Geometric (Fey and Lenssen, 2019). For our independence loss we use a normalized implementation of HSIC provided by Ma et al. (2020) We use the Adam (Kingma and Welling, 2014) optimizer. Experiments have been run on an 11GB GeForce GTX 1080 Ti graphics card. On AO-Clevr we reach our best result in 1-5 epochs on all splits, at ~5 minutes per epoch. On UT-Zappos we reach our best result in ~30 epochs at ~1.5 minutes per epoch. See Appendix A.1 for detailed hyperparameters and grid search ranges.

3.2 Evaluation

We evaluate our approach on two CZSL datasets. Previous works also evaluate on MIT-States (Isola et al., 2015), but Atzmon et al. (2020) conclude through a large-scale user study that the dataset has a level of ~70% label noise, making it too noisy for evaluating compositionality.

AO-Clevr (Atzmon et al., 2020; Johnson et al., 2017) is a synthetic dataset consisting of 3 types of objects (sphere, cube, cylinder) and 8 attributes (red, purple, yellow, blue, green, cyan, gray, brown), with 24 compositional classes in total. There are 6 splits with a varying ratio of unseen to seen classes, ranging from 2:8 to 7:3, allowing insights into the performance of models as the proportion of unseen classes increases.

UT-Zappos (Yu and Grauman, 2014) is a fine-grained dataset of types of shoes with 12 objects (e.g., boat shoes), 16 attributes (e.g., leather), and 116 compositional classes. We use the split proposed by Purushwalkam et al. (2019), with 83 seen classes in the training set, 15 seen and 15 unseen classes in the validation set, and 18 seen and 18 unseen classes in the test set.

Metrics: Like other recent works we adopt the generalized zero-shot evaluation protocol proposed by Chao et al. (2016); Purushwalkam et al. (2019). The main metrics reported, and elaboreted upon below, are:

- Area Under the Seen-Unseen Curve (AUC)
- Closed Seen and Unseen accuracy
- Open Seen and Unseen accuracy
- Harmonic Mean

Instead of evaluating only on the unseen classes as in the Closed setting, the Generalized setting considers both seen and unseen classes in the validation and test sets. To account for the inherent bias towards seen classes, Chao et al. (2016) add a calibration bias term to the activations of unseen classes. As the value of the bias is varied between $-\infty$ and $+\infty$, they draw a curve with the accuracy on the unseen classes on the y axis and the accuracy on the seen classes on the x axis, and report the Area Under the Curve (AUC). The harmonic mean is defined as $2 \cdot \frac{Acc_s \cdot Acc_u}{Acc_s + Acc_u}$, where Acc_s is the accuracy on the seen classes and Acc_u the accuracy on the unseen classes. It penalizes large differences between the two metrics and as such indicates how well the model performs on both seen and unseen classes simultaneously. We also follow prior works in reporting the Closed seen accuracy where only the seen classes are considered (corresponding to a bias of $-\infty$) and closed unseen accuracy where only the unseen classes are considered (corresponding to a bias of $+\infty$). For the results on AO-Clevr we report the seen and unseen accuracy components of the harmonic mean in accordance to Atzmon et al. (2020). Atzmon et al. (2020) do not perform post-hoc bias calibration, but instead handle the seen-unseen bias during training.

Like Naeem et al. (2021), we train our feature extractor end-to-end with the rest of our model. Other methods keep the backbone fixed, but Naeem et al. (2021) have shown that they perform worse when allowed to finetune as they will then start to overfit. While our method is designed to take full advantage of the backbone, we show our results with a fixed backbone in our ablations.

3.3 Results

For each of the results we select the best model by the best harmonic mean on the validation set and report the results for all metrics on the test set. The results with error bars come from 5 training runs with random initializations. We note that, unlike other methods, Causal (Atzmon et al., 2020) does not perform post-hoc bias calibration, but instead tackles the bias against seen classes during training, which is a benefit of their approach.

AO-Clevr Figure 3.1 shows the results on AO-Clevr. We also report the results for CGE (Naeem et al., 2021) (after a grid search using their own codebase) as they have not reported experiments on this dataset. See Appendix A.5 for the exact values and standard error. We observe that for the 3:7, 5:5, 6:4 and 7:3 splits, one or more colors are not part of the training data, making a large portion of the validation and test classes impossible to classify through compositional methods on those splits (cf. Appendix A.3 for more details). Our method performs better than the compared methods on all splits, with minimal impact on the seen accuracy. The improvements are in the range of 2.5 to 20.2% for the harmonic mean and 3.3 to 17.0% for the unseen accuracy, with higher gains as the proportion of unseen classes increases significantly. For a detailed error analysis, cf. Appendix A.4.1.

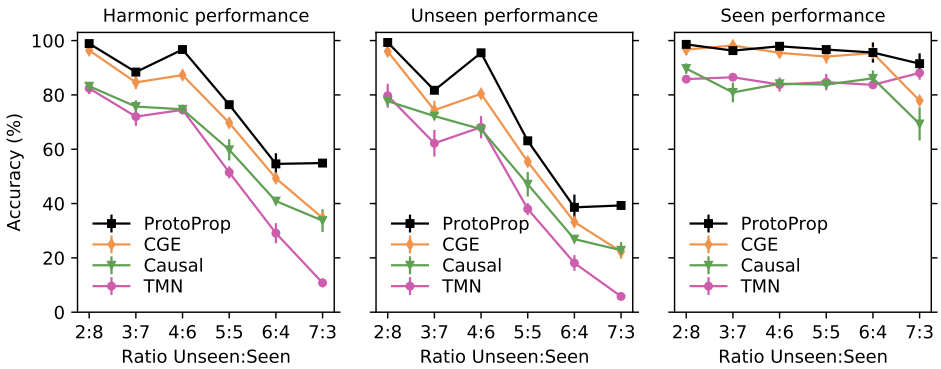

Fig. 3.1 **AO-Clevr results** Plots of the seen and unseen accuracy and their harmonic mean (y-axis) as the ratio of unseen:seen compositional classes (x-axis) increases. ProtoProp consistently outperforms state-of-the-art methods, especially when the portion of unseen classes grows.

Table 3.1 **UT-Zappos results**, ProtoProp improves on state-of-the-art results in the AUC, Closed seen, and harmonic mean metrics.

Method	AUC	Closed seen	Closed unseen	Harmonic
AttOp (Nagarajan and Grauman, 2018)	25.9	59.8	54.2	40.8
LE+ (Misra et al., 2017)	25.7	53.0	61.9	41.0
SymNet (Li et al., 2020)	23.9	53.3	57.9	39.2
TMN (Purushwalkam et al., 2019)	24.7 ± 4.4	58.8 ± 1.4	49.4 ± 4.7	40.7 ± 2.0
Causal (Atzmon et al., 2020)	23.3 ± 0.3	-	55.4 ± 0.8	31.8 ± 1.7
CGE (Naeem et al., 2021)	32.5 ± 1.5	61.0 ± 0.9	$\mathbf{65.9 \pm 1.2}$	47.1 ± 1.7
ProtoProp (ours)	$\mathbf{34.7 \pm 0.8}$	$\mathbf{62.1 \pm 0.9}$	65.5 ± 0.2	$\mathbf{50.2 \pm 1.3}$

UT-Zappos Table 3.1 shows the results on UT-Zappos. Because earlier works only report a single best run, we report the average over 5 random initializations including standard error for the two best performing previous models (CGE and TMN), using their own codebase and reported hyperparameters. We find that, especially for this dataset, the error bars are important, since it is highly susceptible to random initialization; there is a high degree of variability in the harmonic mean for runs with the same hyperparameters, with similar observations for other methods. Our method outperforms earlier methods on all metrics except for the closed unseen accuracy where CGE performs slightly better. Our method performs especially well when both seen and unseen classes are taken into account, as evidenced by the AUC and harmonic mean improvements.

3.4 Ablations

To determine the effect of our visual features in the form of prototypes, in contrast to the semantic features in most prior works, we perform ablations where we replace our prototypes with pretrained word embeddings. We also take a look at the importance of the independence loss function and finetuning the backbone.

Table 3.2 **Ablation on the AO-Clevr 4:6 split**: Checkmarks indicate whether prototypes are used as node features, semantic vectors as node features, local prototypes are trained with independence loss, or the backbone is frozen respectively. Prototypes trained when semantic node features are used are not part of the final classification but still improve the feature extractor output.

Node Features	Indep. Proto	Finetune	Seen	Unseen	Harmonic
Visual		✓	94.5 ± 0.1	77.0 ± 2.8	84.8 ± 1.7
Visual	✓		78.2 ± 1.1	73.0 ± 1.0	75.5 ± 0.8
Visual	✓	✓	$\mathbf{97.9 \pm 1.0}$	$\mathbf{95.5 \pm 0.9}$	$\mathbf{96.7 \pm 0.7}$
Semantic		✓	95.4 ± 1.1	82.4 ± 0.7	88.4 ± 0.1
Semantic	✓	✓	97.3 ± 1.2	84.5 ± 1.4	90.4 ± 0.9

Importance of prototypes Table 3.2 shows the results of our ablations on the 4:6 split on AO-Clevr. Row 1 shows the results for our proposed method without the independence loss function, in row 2 we add the independence loss but freeze the backbone, and row 3 shows the results with both independence and finetuning, as described in the method section. The prototypes perform worse than semantic embeddings when used as node features without the independence loss, but with the independence loss they receive significant gains and come out on top with an 8.3% increase in harmonic mean accuracy over the semantic features. With a frozen backbone the harmonic mean for our method is just slightly higher (0.8%) than Causal on the same split, though it reaches that accuracy in a fraction of the time.

Visual vs. semantic For the ablation in row 4 we use semantic word embeddings (word2vec (Mikolov et al., 2013)) as node features, which is equivalent to Naeem et al. (2021) with a simplified GCN. In row 5 we also train our local prototypes but don't use them during classification, only to influence the local features the feature extractor learns. Training the local prototypes improves the results slightly even when they are not used for the final classification, by encouraging the backbone to learn local features that represent the target attributes and objects.

Effect of independence Table 3.2 shows that the prediction accuracy breaks down when not using the independence loss. Figure 3.2 provides some intuition for this through a t-SNE (Van der Maaten and Hinton, 2008) plot of softmax-pooled attribute (in this case color) representations from our model on AO-Clevr. Without the loss there are three strongly separated clusters per color (one per shape), and many colors are embedded closer to different colors of the same shape. When we do use the independence loss, each of the colors is part of their own single cluster and the only correlations left are those between visually similar colors, improving the accuracy of the compositional prototypes after the propagation step.

3.5 Limitations

Like typical CZSL works, our method is limited to a single attribute and object label per image, and each individual attribute and object needs to be seen in at least one training data point. Existing attribute datasets are often either noisy or do not work in the aforementioned single-attribute setting. But if this initial hurdle of quality datasets is overcome, compositional methods make it easier to handle the addition of new classes, especially for rare objects with

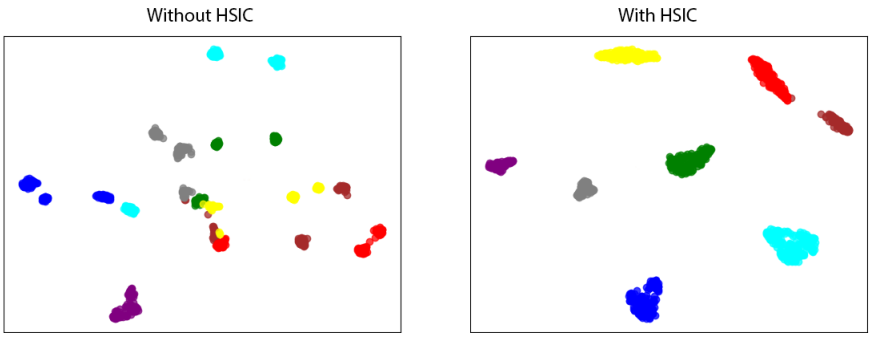

Fig. 3.2 t-SNE plot of softmax-pooled color patches for AO-Clevr that were trained with HSIC loss (left) and without (right), where with the HSIC loss we get a homogeneous grouping of the attributes and no fragmentation by spurious confounders.

little to no available images. We address these concerns in the following chapter, where we extend our method to the multi-attribute setting.

The hyperparameters can be difficult to tune, as there are hyperparameters for the backbone, prototype layer, independence loss, and compositional GNN. We do find, however, that the prototype layers can be tuned separately from the rest, as the best performing individual prototypes and the hyperparameters that led there also perform best when used in conjunction with the GNN. As such, most hyperparameters can be adopted from existing prototype-based classification works that have been trained on a similar dataset.

Chapter 4

Multi-attribute extension

Up to this point our method and evaluation setting was limited to the Compositional Zero-Shot Learning (CZSL) setting, where each image contains exactly one object and one attribute label. In this chapter, we extend our method to a more complex setting, with an arbitrary number of attribute labels per image. Instead of certain compositions, as in the CZSL setting, entire classes are now unseen.

4.1 Method

We again have a set of images X and their class labels $Y = Y_s \cup Y_u$, and we assume access to an attribute matrix $A \in \mathbb{R}^{|Y| \times n}$, where for each class we have a row of n attributes with a real-numbered value $k \in [0, 1]$ (cf. Figure 4.1). The attribute values denote how often the corresponding attribute was identified in an image belonging to the corresponding class, with 0 being never and 1 being 100% of the time. In practice these matrices are often very sparse, only a small number of attributes is observed for any specific class. From a graph perspective, each row of A corresponds to a compositional class node, each column to an attribute node, and each nonzero entry to an edge weight. Importantly, we no longer need to separate our representations into attributes and objects. One way the multi-attribute extension could have worked, was to split the visual primitives into even more groups than attributes and objects, where all mutually exclusive attributes are clustered in the same group. However, this is not scalable, which is why the CZSL setting is limited to shapes in the form of objects, and visual variations of those shapes in the form of attributes. With a multi-label approach we are no longer bound by this limitation, and can group everything into one set of visual primitives, which we now simply call attributes.

Prototype layer To extend the prototype layer from Section 2 to the multi-attribute setting, we make a couple of simple changes, as shown in Figure 4.2. First, the prototype compatibility score s_a is recalculated over the attention-pooled input, instead of taking the maximum compatibility score over all patches. Second, we find that, due to the sparsity of A, regression and multi-label loss functions tend to collapse to either 0 or the mean, when training prototypes on the attribute labels. Instead, we calculate a class compatibility score s_{pc} through a matrix-vector product:

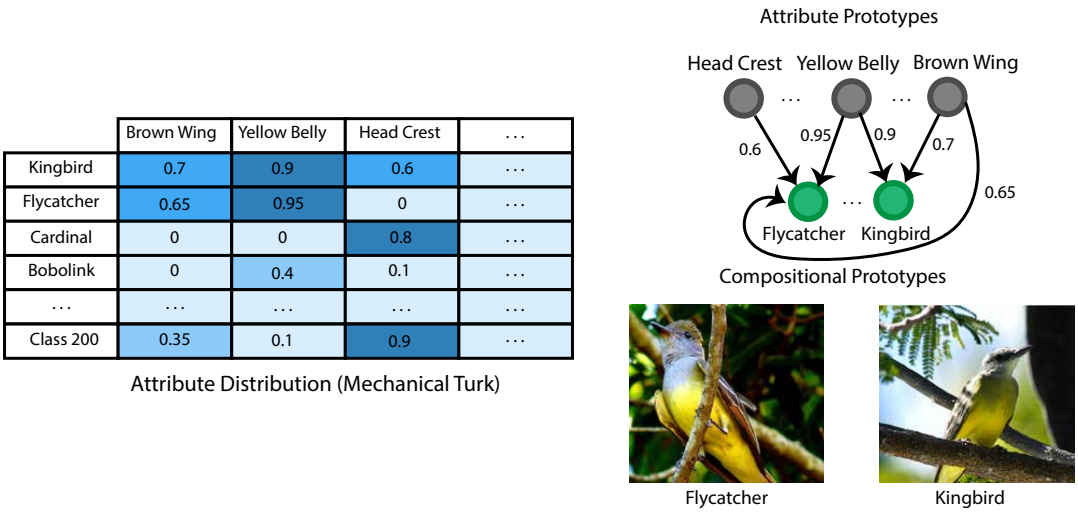

Fig. 4.1 **Multi-attribute extension** In this setting, we simplify our graph (single layer, no learnable weights) of attributes and compositional prototypes, weighted by an attribute distribution that is calculated by annotations from mechanical turk workers. Unlike before, arbitrary number of attributes can now appear in a single image.

$$\hat{A} \cdot s_a^T = s_{pc} \in \mathbb{R}^{|Y|} . \tag{4.1}$$

Here $\hat{A}$ is a row-wise L1-normalized version of A, to ensure that classes with more attributes do not have a higher score by default. Through this equation, s_{pc} is an aggregation each of the attribute scores, normalized and weighted by the prior knowledge of the attribute weight distribution for each class. We optimize s_{pc} through a cross-entropy loss function, which encourages the network to lower the scores for attributes that should not be indicative of the target class, and increase the scores of those that should. This provides a strong gradient to each of the individual prototypes, without risk of collapse to a trivial solution.

Compositional prototypes Our compositional prototypes P_c are constructed in a similar way as in Equation 4.1:

$$\hat{A} \cdot P_a = P_c \in \mathbb{R}^{|Y| \times C} . \tag{4.2}$$

Here P_a is the set of attribute prototypes from our prototype layer, and C is the prototype dimension. Importantly, there is no further transformation of the attribute prototypes, and only a single propagation and aggregation step. We find that more complex graph structures and multi-layered GNNs reach similar results, but here we have reported the simplest method that works at least as well. Just like in Section 2.3, we compute a global class compatibility score s_c, through dot product between our prototypes P_c and the average-pooled backbone output, and classify input samples with $y = \text{argmax}(s_c)$.

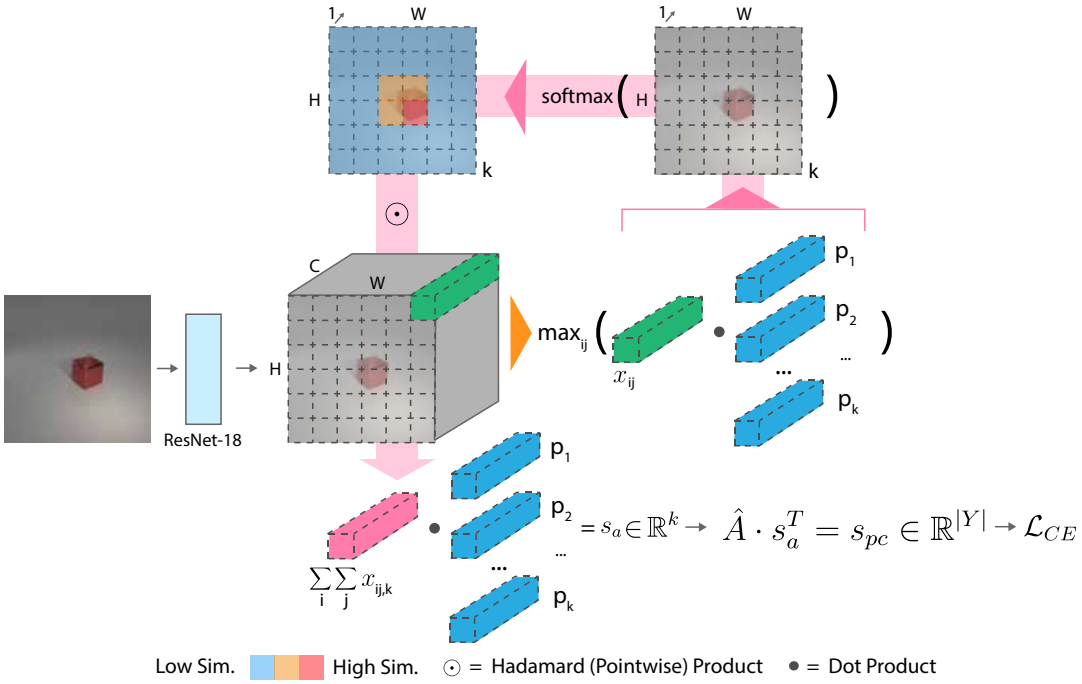

Fig. 4.2 **Multi-Label Prototypes** The prototype compatibility scores s_a are calculated by a dot product between the attention-pooled input and prototypes. The prototype scores are then multiplied by the attribute matrix A, resulting in the class compatibility scores s_{pc}.

Independence

Unlike in the CZSL setting, decorrelation and independence losses do not improve the results on the multi-attribute datasets we have experimented with. In the CZSL setting, the datasets are significantly underspecified; some attributes are only ever seen on one or two object types, and often only have a small number of example images. In the case of the multi-attribute datasets, however, this is no longer the case. In the Caltech-UCSD Birds (CUB) dataset, e.g., each individual attribute is recognizable, to some extent, in at least 4 and on average 70 different classes, out of 200 classes total. This already provides regularization through sample diversity in both bird species and their environment, where the model can't overfit on spurious characteristics of a single class or location. Furthermore, looking at the attribute distributions of the training data and the test data separately in Figure 4.3, we can see there is no significant difference.

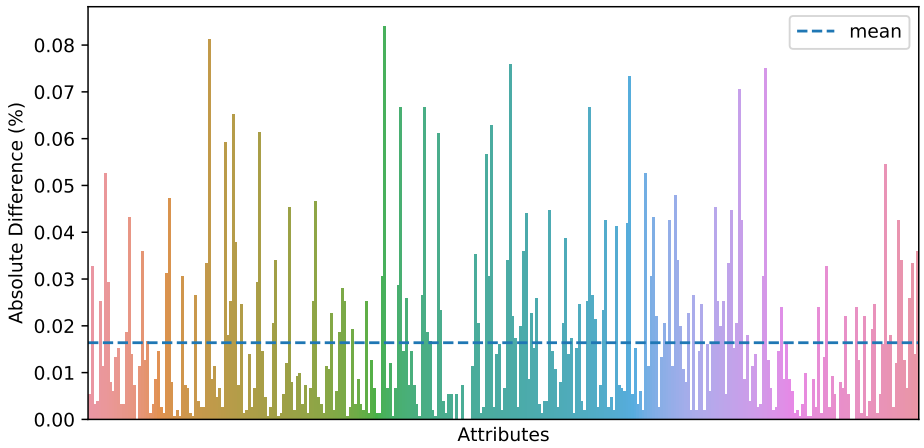

Fig. 4.3 **Train-test attribute distribution difference** Here, the maximum absolute difference in the distribution of the attributes in the seen attributes $a_s \in A_s$ vs. the unseen attributes $a_u \in A_u$ is only 8%, with an average of 1.6%.

Xu et al. (2020) use a decorrelation loss to show that, while the loss does not affect the classification accuracy, it does improve part *localization*, e.g., by discouraging the network to share features between a bird's white belly and a white breast.

4.2 Experiments

4.2.1 Evaluation

We evaluate our approach on two common zero-shot learning datasets, CUB (Wah et al., 2011) and Animals with Attributes 2 (AWA2) (Xian et al., 2018). Both datasets come with an attribute matrix A, as described in Section 4.1. In line with related work, our backbone is a ResNet-101.

CUB CUB is a dataset consisting of 11.788 images of 200 fine-grained bird species. There are 312 attributes in total, including shapes such as 'pigeon-like', part colors such as 'red crown', and wing patterns such as 'stripes'. The dataset is split into a training set containing 100 classes, a validation set containing 50 classes, and a test set containing 50 classes, with no overlap between sets.

AWA2 The original Animals with Attributes dataset (Lampert et al., 2009) was suspended due to copyright issues, and has now been followed up by AWA2. The dataset consists of 37.322 images of 50 animal classes such as bears, whales, and tigers. There are 85 attributes, including environment such as water, diet such as carnivore, and appearance such as furry. The dataset is split into a training set containing 27 classes, a validation set containing 13 classes, and a test set containing 10 classes, with no overlap between sets.

Metrics Just like in the CZSL setting, our main metric is the harmonic mean of the open seen and unseen accuracies (i.e., both seen and unseen classes are considered during evaluation). We also report the open unseen and seen accuracy, and again add a calibration bias term to the activations of unseen classes.

We note that all related works we compare to combine the training and validation sets into one large training set, and validate and select their models on the test set, which allows overfitting on the test set. Results including a standard deviation have been calculated over 5 random initializations.

4.2.2 Results

Table 4.1 shows the results on CUB and AWA2. It seems that, with the given attribute vectors, the zero-shot performance on both datasets has mostly reached a maximum. On CUB, AREN, RGEN, APN, and our method all perform on par in terms of harmonic mean, with some variation in the unseen and seen accuracies. The same is mostly true for AWA2, though RGEN does reach a significantly higher unseen accuracy than other methods, while ProtoProp is better at preserving a high accuracy on the seen data. Many differences between similar species of birds are so fine-grained that the attribute vectors alone are not sufficient to differentiate them, cf. Appendix A.4.3 and A.4.4 for a detailed error analysis.

Table 4.1 **Multi-attribute results**

	CUB			AWA2		
Method	Unseen	Seen	Harmonic	Unseen	Seen	Harmonic
LDF (2018b)	26.4	81.6	39.9	9.8	87.4	17.6
SGMA (2019)	36.7	71,3	48.5	37.6	87.1	52.5
LFGAA (2019a)	36.2	80.9	50.0	27.0	93.4	41.9
AREN+CS (2019)	69.0	63.2	66.0	54.7	79.1	64.7
APN (2020)	65.3	69.3	67.2	56.5	78.0	65.5
RGEN (2020)	60.0	73.5	66.1	64.1	76.4	69.7
ProtoProp (Ours)	62.5 ± 1.7	72.8 ± 1.8	67.2 ± 0.2	57.4 ± 0.7	86.0 ± 0.9	68.9 ± 0.7

Ablation Table 4.2 shows ablation results, including standard deviation over 5 random initializations. The best harmonic mean when using Equation 4.1, i.e., the combined score calculated by each of the attribute prototypes separately, is significantly lower than calculating the scores using the compositional prototypes, indicating that the model learns compositional prototypes that perform better than the sum of their parts. The model performs slightly better when using the attention-pooled output to calculate the final prototype scores. Unlike in the CZSL setting, where the attention weights only attend to a single patch, in the multi-attribute setting the weights are more evenly spaced over multiple patches. As such, the attention-pooled output can provide a more accurate score than only looking at the patch with the maximum compatibility score.

Random attribute variation In this experiment, we measure the effect of randomly varying the attribute labels of the CUB dataset, as a proxy to annotation error or distribution shift. In Figure 4.4, we randomly vary the attribute values $a \in A$, proportional to the original value, at

Table 4.2 **Ablation**

Attention	Compositional	Harmonic
	✓	64.3 ± 0.6
✓		61.44 ± 0.8
✓	✓	67.2 ± 0.2

the rate given on the x-axis. E.g., for each rate r we calculate a new attribute value $a = a \cdot (1+e)$, with $e \in E \sim U(-r, r)$. The model is fairly robust to these variations, with harmonic mean dropping slowly at first, just 1 or 2 percentage points per 10% increase in random variation, and more steeply around $r = 0.4$ onwards. The model reaches a harmonic mean of 51.6% at $r = 0.6$ vs. 67.2% at $r = 0$. The difference between the seen and unseen attribute distributions, as discussed in Section 4.1, is roughly equivalent to $r = 0.1$, providing further evidence that this distribution shift does not significantly affect performance.

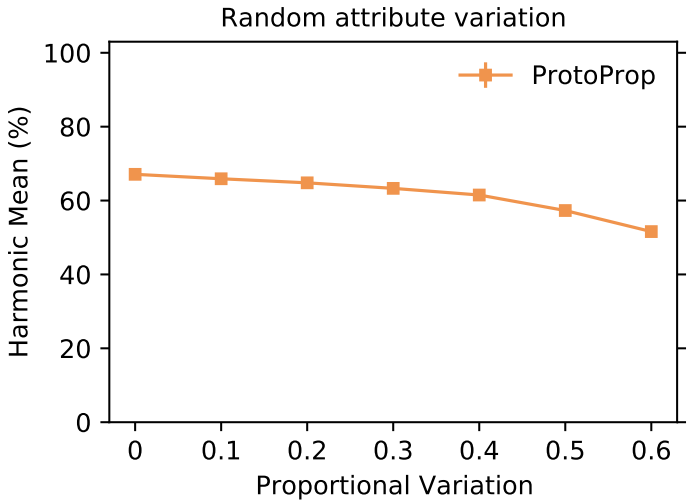

Fig. 4.4 **Random attribute variation on CUB** We randomly vary the attribute values $a \in A$, proportional to the original value, at the rate given on the x-axis. E.g., for each rate r we calculate a new attribute value $a = a \cdot (1 + e)$, with $e \in E \sim U(-r, r)$. We find that the accuracy is fairly robust to these random variations, the harmonic mean drops just 1 or 2 percentage points for every 10% increase in random variation.

4.3 Limitations

The main limitation of the multi-attribute setting, specifically datasets dataets such as CUB and AWA2, is the class-level attribute distribution matrix A. Each image is annotated with the same attribute labels, regardless of if the attributes are visible in a given image. As a result, the training is more noisy, and especially rare exceptions, such as a species looking significantly different when young vs. mature, will become difficult to recognize. A benefit of this approach, though, is that the annotation process is much less labour-intensive, as only a subset of images needs to be annotated to estimate a global attribute distribution.

One way to (partly) remedy the aforementioned limitation may be to, e.g., employ a neural prototype tree approach, such as a zero-shot adaptation of Nauta et al. (2021). An early example of such an approach, sans prototypes, is the zero-shot random forest based on relative attributes by Cheng et al. (2017), which would be interesting to revisit with a stronger attribute localization method. Such a model could, if trained properly, learn separate branches for a young and a mature bird of the same species, to ensure that irrelevant attributes do not contribute to the classification score.

To reach the maximum accuracy on both of the evaluated datasets, we need a prototype dimension of 1024, significantly higher than what was necessary in the CZSL setting. Especially on datasets like CUB, with a large number of attributes, this will result in a relatively high memory usage.

Finally, as discussed in Section 4.2.2, it seems that a performance ceiling has been reached for the datasets we have evaluated. This makes it difficult to compare which methods truly perform best. In future work, it would be interesting to apply the method to a more complex dataset, such as GQA (Hudson and Manning, 2019), a large visual question answering and object detection dataset.

4.4 Conclusion

The multi-attribute extension effectively deals with the single-label restrictions from the Compositonal Zero-Shot Learning (CZSL) setting, and reaches accuracy on par with state-of-the-art methods on two challenging multi-attribute zero-shot benchmarks. The performance on the datasets evaluated in this paper seems to have reached a ceiling; most of the errors made can be attributed to a lack of distinguishing information in the attribute vectors, and progress on these datasets has been stagnant in the last 2 to 3 years. In future work, it would be interesting to apply the method to a more complex dataset, such as GQA (Hudson and Manning, 2019), a large-scale visual question answering and object detection dataset.

References

Yuval Atzmon, Felix Kreuk, Uri Shalit, and Gal Chechik. A causal view of compositional zero-shot recognition. *Advances in Neural Information Processing Systems*, 33, 2020.

Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European conference on computer vision*, pages 52–68. Springer, 2016.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32:8930–8941, 2019.

Yuhu Cheng, Xue Qiao, Xuesong Wang, and Qiang Yu. Random forest classifier for zero-shot learning based on relative attribute. *IEEE transactions on neural networks and learning systems*, 29(5):1662–1674, 2017.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

Arthur Gretton, Kenji Fukumizu, Choon Hui Teo, Le Song, Bernhard Schölkopf, Alexander J Smola, et al. A kernel statistical test of independence. In *Nips*, volume 20, pages 585–592. Citeseer, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Martin Hebart, Charles Y Zheng, Francisco Pereira, and Chris Baker. Revealing the multidimensional mental representations of natural objects underlying human similarity judgments. 2020.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=8E1-f3VhX1o.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019.

Phillip Isola, Joseph J Lim, and Edward H Adelson. Discovering states and transformations in image collections. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1383–1391, 2015.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9012–9020, 2019.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE, 2009.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018a.

Yan Li, Junge Zhang, Jianguo Zhang, and Kaiqi Huang. Discriminative learning of latent features for zero-shot recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7463–7471, 2018b.

Yong-Lu Li, Yue Xu, Xiaohan Mao, and Cewu Lu. Symmetry and group in attribute-object compositions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11316–11325, 2020.

Yang Liu, Jishun Guo, Deng Cai, and Xiaofei He. Attribute attention for semantic disambiguation in zero-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6698–6707, 2019a.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2537–2546, 2019b.

Kurt Wan-Duo Ma, J. P. Lewis, and W. Bastiaan Kleijn. The HSIC bottleneck: Deep learning without back-propagation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5085–5092. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5950.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119, 2013.

Ishan Misra, Abhinav Gupta, and Martial Hebert. From red wine to red tomato: Composition with context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1792–1801, 2017.

Muhammad Ferjad Naeem, Yongqin Xian, Federico Tombari, and Zeynep Akata. Learning graph embeddings for compositional zero-shot learning. *arXiv preprint arXiv:2102.01987*, 2021.

Tushar Nagarajan and Kristen Grauman. Attributes as operators: factorizing unseen attribute-object compositions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 169–185, 2018.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Senthil Purushwalkam, Maximilian Nickel, Abhinav Gupta, and Marc'Aurelio Ranzato. Task-driven modular networks for zero-shot compositional learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3593–3602, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Ruslan Salakhutdinov, Antonio Torralba, and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR 2011*, pages 1481–1488. IEEE, 2011.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of machine learning research*, 2(Nov):67–93, 2001.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6861–6871. PMLR, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

Guo-Sen Xie, Li Liu, Xiaobo Jin, Fan Zhu, Zheng Zhang, Jie Qin, Yazhou Yao, and Ling Shao. Attentive region embedding network for zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9384–9393, 2019.

Guo-Sen Xie, Li Liu, Fan Zhu, Fang Zhao, Zheng Zhang, Yazhou Yao, Jie Qin, and Ling Shao. Region graph embedding network for zero-shot learning. In *European Conference on Computer Vision*, pages 562–580. Springer, 2020.

Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. Attribute prototype network for zero-shot learning. In *34th Conference on Neural Information Processing Systems*. Curran Associates, Inc., 2020.

Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *In ICML Workshop on Deep Learning*. Citeseer.

Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014.

Yizhe Zhu, Jianwen Xie, Zhiqiang Tang, Xi Peng, and Ahmed Elgammal. Semantic-guided multi-attention localization for zero-shot learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Appendix A

A.1 Hyperparameters

AO-Clevr Like Atzmon et al. (2020) we performed grid searches over the following splits: {2:8, 5:5, 6:4, 7:3}. We used the largest batch size that could fit in memory on our limited hardware, which was 256 for an image size of 224x224. For the learning rate (Adam (Kingma and Welling, 2014) optimizer) we searched in the range of {0.001, 0.0001, 1e04, 5e-4, 5e-5}, with weight decay {0, 5e-4. 5e-5}. We chose a weight decay of 5e-5 and learning rate of 5e-4 until the 4:6 split and 1e-4 afterwards.

Prototype dimension: {256, 300, 512}, backbone output dimension: {256, 300, 512}, Graph layers: {1, 2, 3}, graph hidden dimension: {256, 512, 1024, 2048, 4096}, λ_h: {0, 1, 5, 10, 25, 50}, Clst: {0, 0.1, 0.05, 0.01, 0.005}, Sep: {0, 0.1, 0.05, 0.01, 0.005}.

We chose a prototype dimension of 256, backbone output of 512, 2 graph layers, graph hidden dimension of 512, λ_h of 10, Clst and Sep of 0.01.

UT-Zappos we again used the Adam optimizer, with learning rate in the ranges {5e-5, 5e-4, 5e-3}, and weight decay {0, 5e-4. 5e-5}, where we chose a learning rate and and weight decay of 5e-5 and a batch size of 128. For the rest of the parameters we searched the same ranges as above, where the same choices were optimal as for AO-Clevr.

A.2 Hilbert-Schmidt Independence Criterion

The (biased) empirical HSIC estimator (Gretton et al., 2007) is defined as:

$$\mathrm{HSIC}(U, V) = \frac{1}{m^2}\,\mathrm{trace}(\mathbf{KHLH})$$

Where $\mathbf{K}$ and $\mathbf{L}$ are $m \times m$ matrices with entries k_{ij} and l_{ij}, $\mathbf{H} = \mathbf{I}\frac{1}{m}\mathbf{1}\mathbf{1}^\top$, and $\mathbf{1}$ is a $1 \times m$ vector of ones. The elements of $\mathbf{K}$ and $\mathbf{L}$ are outputs of a kernel function over the inputs U and V such as the (universal) gaussian kernel $k_{ij} := \exp\left(-\sigma^{-2}\left\|u_i - u_j\right\|^2\right)$ where σ is the kernel size. We follow Gretton et al. (2007) in setting the kernel size to the median distance between points, but universality of the gaussian kernel holds for any kernel size. The empirical estimator has a bias in the order of $O(m^{-1})$ which is negligible at even moderate sample sizes.

A.3 Dataset Issues

A.3.1 AO-Clevr

Some of the attributes are not part of the training data for several unseen:seen splits, violating an important CZSL assumption, and making it impossible to recognize certain compositions. Specifically, for the 'unseen:seen-n_seed' seeds, the following fractions of unseen test samples are impossible to compose due to missing attributes:

- $\frac{900}{2400}$ for 3:7-0 and 3:7-1.
- $\frac{900}{3600}$ for 5:5-0 and 5:5-2.
- $\frac{2700}{4500}$ for 6:4-0 and 6:4-1, $\frac{1800}{4500}$ for 6:4-2.
- $\frac{900}{5100}$ for 7:3-0, $\frac{2700}{5100}$ for 7:3-1 and 7:3-2.

A.3.2 UT-Zappos

UT-Zappos is a dataset of fine-grained types of shoes with material labels, where the material labels can not necessarily always be seen as a transformation that is grounded in vision. Many labels mostly serve for humans to inform the material the shoes are made of, but are visually indistinguishable from other shoes with different material labels (cf. Figure A.1). 8 our of 16 materials are some type of leather, and 3 materials are specifically designed to resemble other materials as much as possible (faux fur, faux leather, synthetic). Compound materials are only labeled with a single material, and not necessarily with the most prevalent material. Colors and patterns, combined with the low resolution of the images, often obscure crucial details and textures required to recognize certain materials.

Fig. A.1 **UT-Zappos**, examples of visually meaningless material labels.

A.4 Error Analysis

A.4.1 AO-Clevr

Aside from color and shape labels, AO-Clevr also provides size (small or large) and material (rubber or metal) labels. We look at the 4:6 split, where 47.0% of all test samples are labeled 'small' (and 53.0% 'large'), and 52.2% are labeled 'rubber' (and 47.8% 'metal'). For the default image size of 96×96, 82.1% of all errors are made on test samples that are labeled 'small', indicating a significant limitation in recognizing small objects. Rubber vs metal, or rather

matte vs shiny, does not offer significant difficulties, as 50.9% of errors are made on samples labeled 'rubber', which is close to the expected error distribution.

To increase the performance on small objects, we look at two straightforward options: multi-scale architecture (Ronneberger et al., 2015), and simply increasing the input image size. Yosinski et al. have shown that neural networks learn more general, fine-grained features, such as corners or eyes, in their earlier layers, and global features representing entire objects in their final layers. A multi-scale architecture takes advantage of this fact by leveraging the outputs of multiple layers, instead of only the final layer. Figure A.2 shows our multi-scale architecture. We repeat our prototype layer at 3 different output layers of the backbone, and sum the output logits for the final classification score. This approach brings the error on 'small' samples down to 67.2% (a 14.9% absolute improvement), while also improving the final classification score.

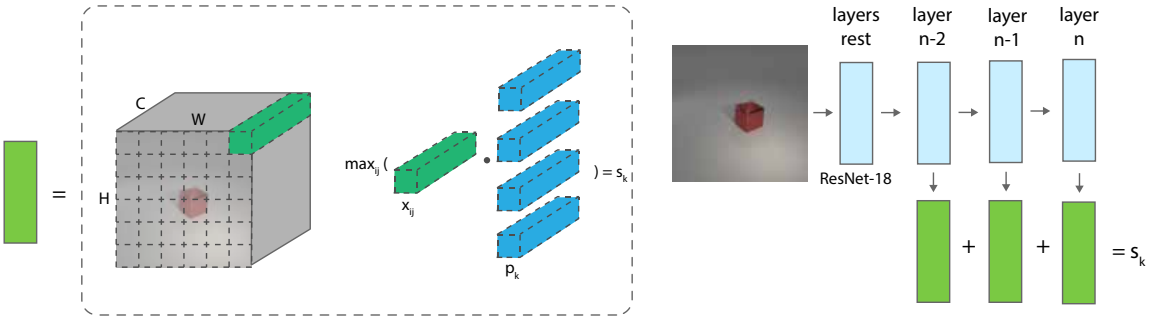

Fig. A.2 **Multi-Scale architecture**, prototype layers are placed at multiple scales in the backbone, their compatibility scores are combined for the final prediction.

Simply increasing the input image size to 224×224, however, leads to a much greater improvement, with a test sample error on 'small' objects of just 56.9% (a 10.3% absolute improvement over multi-scale). Both approaches significantly increase memory requirements, the multi-scale approach slightly less than increasing the image size. The multi-scale approach requires more computational resources, as the extra prototype layers require calculating the independence loss and spatial attention weights multiple times. Even though the final error of 56.9% (as opposed to the expected 47.0%) still indicates some difficulty with recognizing small objects, utilising both approaches at the same time does not improve the final accuracy.

Finally, Figure A.3 shows confusion matrices for the attribute and object predictions. We can see that almost all attribute errors misclassify a purple object as a red object. This can be explained by the fact that purple and gray are the only colors that appear in just one compositional class in the training data, and purple is close to red in RGB values. For the objects, cubes and spheres are never confused, but cubes and spheres are sometimes misclassified as cylinders and vice versa. This can be explained by the fact that cylinders combine properties of spheres and cubes (both a flat and a rounded side). The error of confusing spheres for cylinders is most prevalent, as the training data contains twice as many cylinders and cubes as spheres. As the unseen:seen split increases we can see these kinds of errors exacerbate; blue and cyan start being confused, as well as red, brown, and yellow.

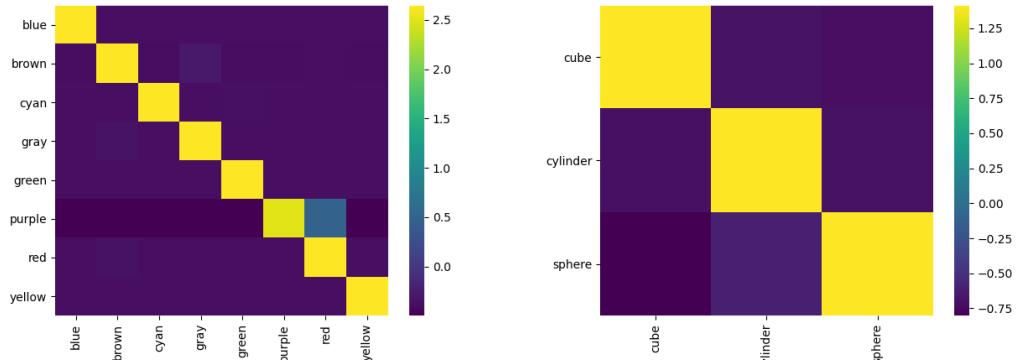

Fig. A.3 **AO-Clevr confusion matrices** for the 4:6 split.

A.4.2 UT-Zappos

Figure A.4 shows confusion matrices for the attribute and object predictions on UT-Zappos. For the object classifications most errors are fairly straightforward; knee-high boots, e.g., are confused with mid-calf boots, since the white background and cropping of the images often makes it impossible to infer scale. Boat shoes and slippers often have exactly the same shape as certain loafers.

The same can be seen for the attribute predictions, but as explained in Appendix A.3, the labels here are often visually indistinguishable. Faux leather, a material that has been explicitly designed to look as much like real leather as possible, is almost exclusively misclassified as real leather. Full-grain leather is often visually indistinguishable from regular leather. Suede and Nubuck are both a type of sanded leather, which are difficult to distinguish when colored and in low resolution images. Synthetic and Cotton shoes, again, attempt to emulate different material types (cf. Figure A.1 for examples). Most of the errors are therefore mainly caused by attribute misclassifications; 59.7% of the errors have the correct object label, 11.4% have the correct attribute label, and the remaining 28.9% misclassify both.

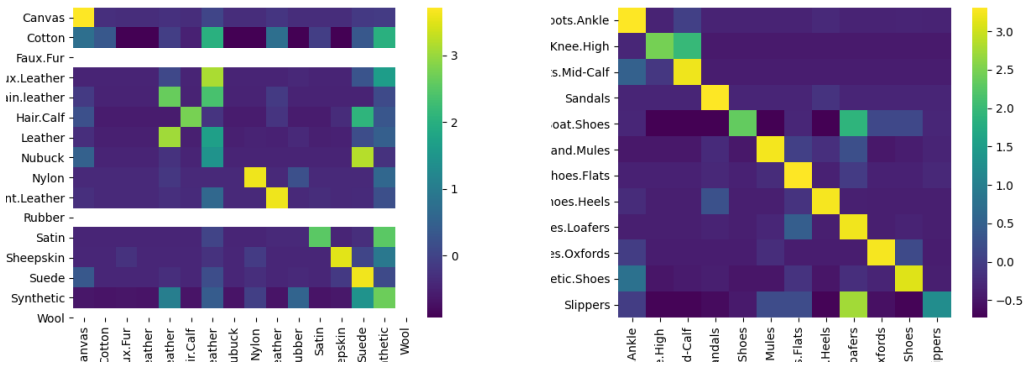

Fig. A.4 **UT-Zappos confusion matrices**, Faux.Fur, Rubber, and Wool are not part of the test set, and therefore displayed blank.

A.4.3 CUB

Figure A.5 shows a confusion matrix of the classification accuracy of our best performing model on CUB (zoom in for more details). Most errors occur between similar looking birds in the same subfamily, such as terns or sparrows. The errors are mostly distributed in the same column, i.e., in a set of similar looking birds, there is one bird that is predicted most of the time. Take, e.g., two birds that have a yellow belly and green wing that are visible in most of their pictures, but the second bird has other prominent features that are only visible some of the time. The first bird will have a higher weight for the yellow belly and green wing attributes, since it has fewer other defining attributes. As such, the second bird will be classified as the first bird in most of its pictures.

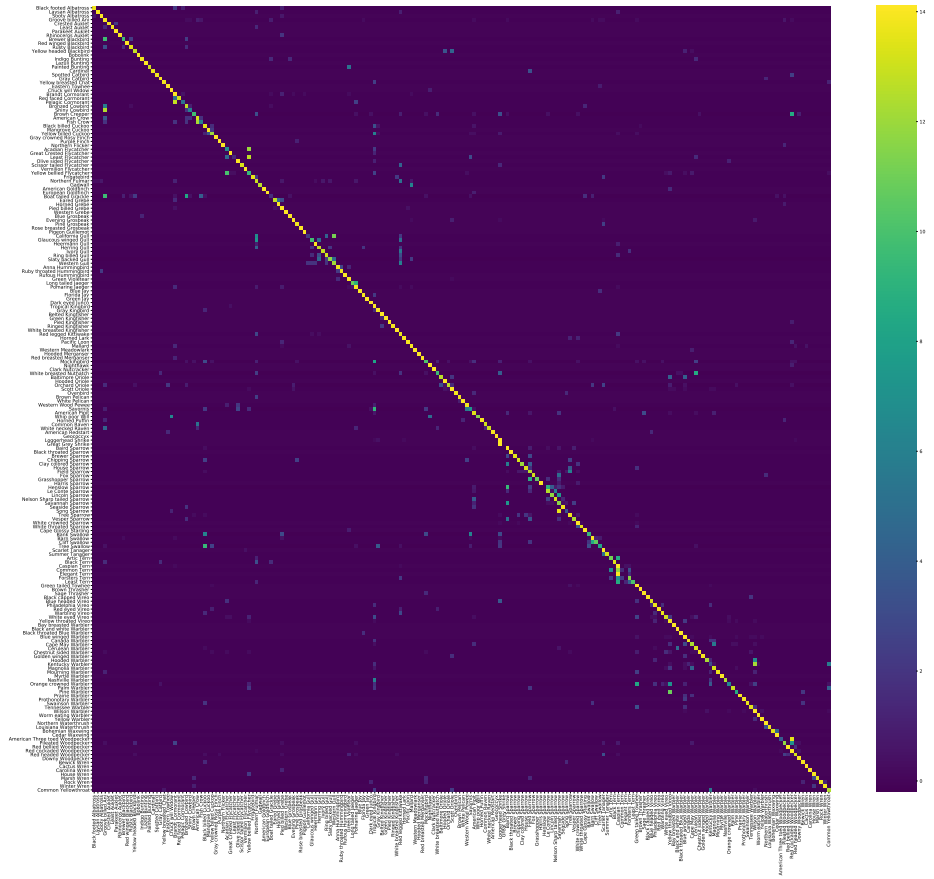

Fig. A.5 **CUB Confusion Matrix** (vector image, zoom for more details)

Figure A.6 shows a heatmap of cosine similarity between attribute vectors. The vast majority of errors occurs between classes that have a cosine similarity between attribute vectors greater than 0.95, with the vectors for fish crows and American crows being almost equal at 0.996.

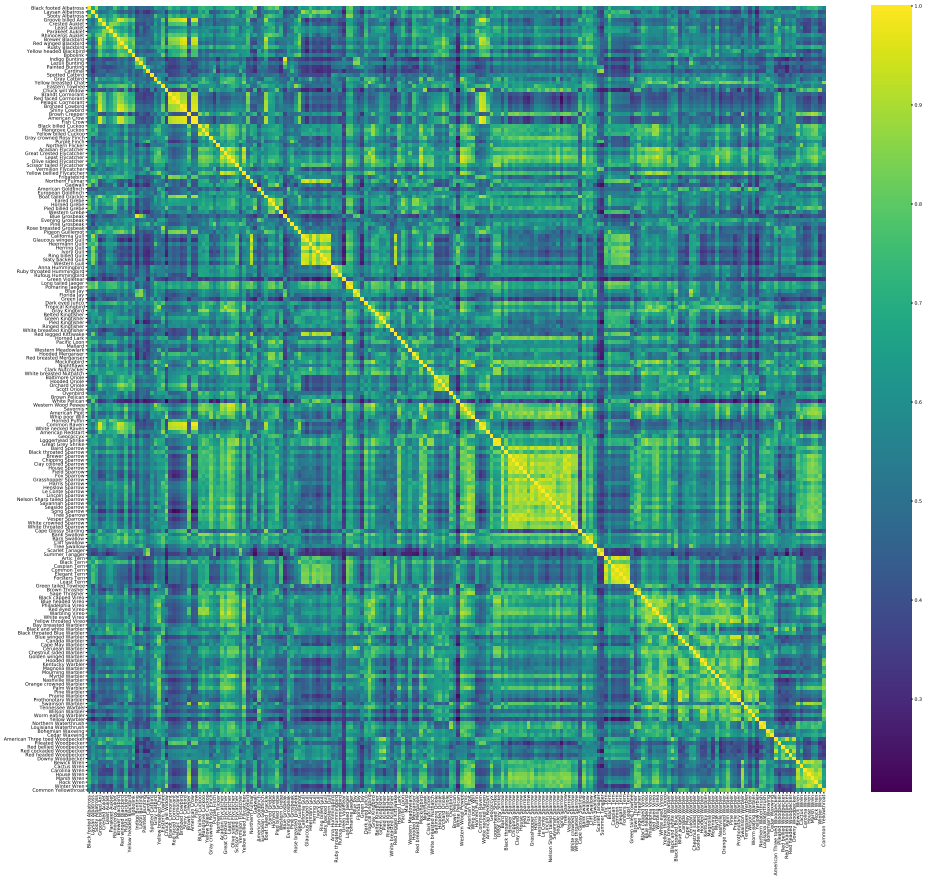

Fig. A.6 **CUB Class Similarity Matrix** (vector image, zoom for more details)

Figure A.7 shows examples of the most common errors on CUB. On the top row, left and middle, we have a green-tailed towhee. On the top row, right, an orange-crowned warbler. Here the leftmost image is misclassified as an orange-crowned warbler, even though their attribute vectors only have a cosine similarity of 0.79. Certain obstructions or angles hide defining features, making the bird look more like a different, generally dissimilar, species. On the bottom row, we have caspian, common, and elegant terns. Here, the attribute matrices alone are not enough to differentiate caspian and elegant terns. The common tern has different colored feet (orange vs black) as its most distinguishing feature, but their feet are retracted into their feathers during flight.

Other notable errors occur when, e.g., the bird significantly varies in appearance between age and gender. Some sayornis, e.g., might have yellow-feathered bellies, even though the majority of the labeled images do not, leaving it with a low attribute score for 'yellow belly'. In those cases, the sayornis are misclassified as a tropical kingbird, which has a prominent yellow belly.

Fig. A.7 **Common errors** top row, left and middle: Green tailed Towhee, top row, right: Orange crowned Warbler. Bottom: Caspian, Common, and Elegant Terns.

It would be interesting to investigate the accuracy of the individual attribute prototypes, but since the attribute labels are on the class level instead of image level, that would require manual inspection of each image, which is too labour-intensive for now.

A.4.4 AWA2

While animals with attributes 2 (AWA2) is much less fine-grained than the Caltech Birds (CUB) dataset, the attribute matrix is much less fine-grained as well. As such, we see similar errors here, that are mainly caused by a lack of fine-grained description.

Sheep and horses, for example, are almost exclusively classified as cows (cf. Figure A.8). The attribute vectors of sheep and cow have a cosine similarity of 0.85, horse and cow 0.86. While that seems like there should be a large enough difference to be distinguishable, the only attributes where they significantly differ are domestic, mountains, small, and big. In the images, sheep, horses, and cows predominantly appear on a green field with similar levels of incline, cropped in such a way that there is no sense of scale, are domestic, and are either white, black, brown, or a combination thereof. As such, none of the attributes with a significant difference are actually useful as a visual distinction. Blue whale and humpback whale have a cosine similarity of 0.96, and over 95% of their pictures are just their tail fin above water, making the vast majority of attributes inapplicable. According to the attribute matrix, the most distinguishing attribute for bats w.r.t. the other rodents or small animals is that they are cave animals, but none of the pictures of bats are taken in caves. As such, bats are, e.g., classified as hamsters when they are held in a hand, as spider-monkeys when they hang from a branch, and as mice when they are surrounded by foliage.

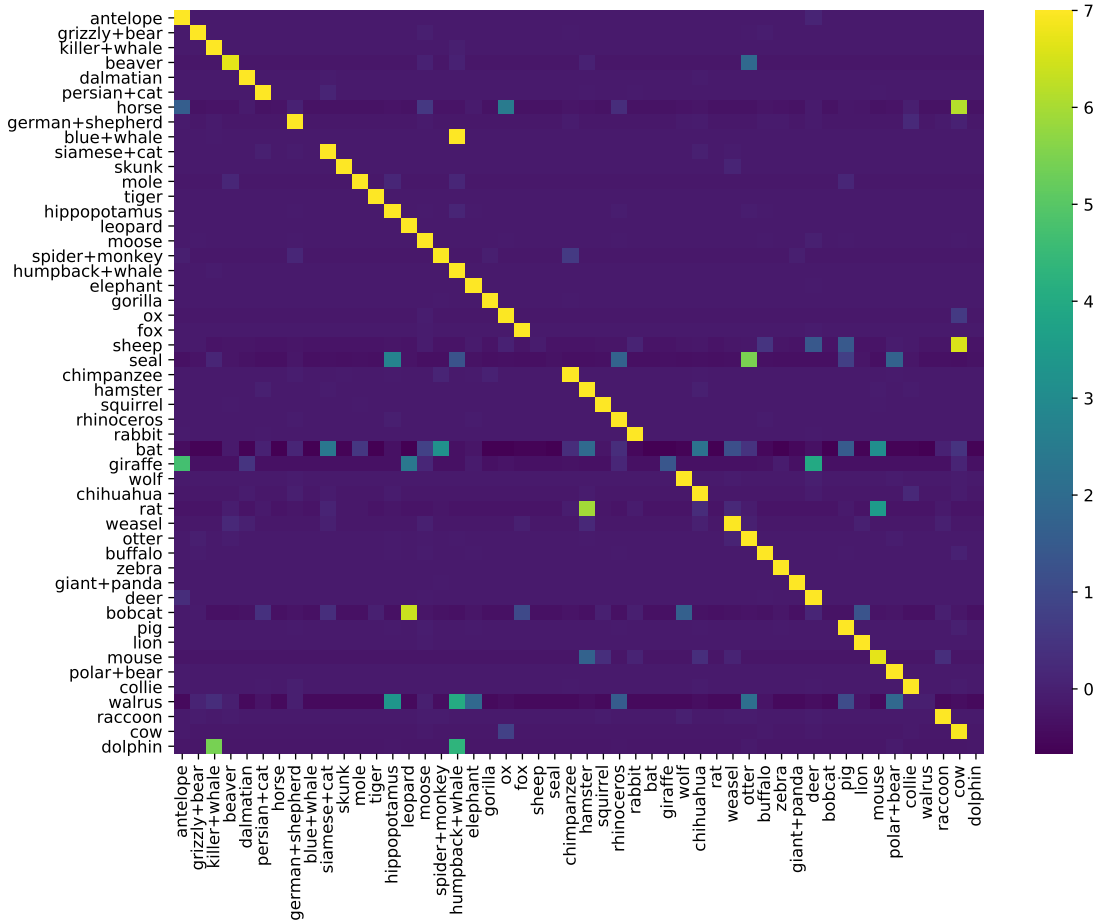

Fig. A.8 **AWA Confusion Matrix** (vector image, zoom for more details)

A.5 AO-Clevr Results

Table A.1 **AO-Clevr Results**: ProtoProp consistently outperforms state of the art methods, especially when the portion of unseen classes grows.

	ProtoProp (ours)			Causal		
U:S	Seen	Unseen	Harmonic	Seen	Unseen	Harmonic
2:8	98.6 ± 0.6	99.3 ± 0.4	98.9 ± 0.3	89.7 ± 1.9	77.7 ± 1.4	83.2 ± 1.2
3:7	96.3 ± 0.9	81.7 ± 0.1	88.4 ± 0.4	80.9 ± 3.6	72.2 ± 1.0	75.7 ± 2.3
4:6	97.9 ± 1.0	95.5 ± 0.9	96.7 ± 0.7	84.1 ± 1.8	67.4 ± 2.0	74.7 ± 1.7
5:5	96.7 ± 0.2	63.1 ± 0.4	76.4 ± 0.3	83.8 ± 0.8	47.1 ± 4.5	59.8 ± 3.9
6:4	95.6 ± 3.7	38.6 ± 4.7	54.6 ± 3.9	86.1 ± 2.9	26.9 ± 0.5	40.9 ± 1.0
7:3	91.5 ± 4.6	39.3 ± 1.6	54.9 ± 0.6	69.3 ± 6.1	22.8 ± 3.0	33.7 ± 4.2

	CGE			TMN		
U:S	Seen	Unseen	Harmonic	Seen	Unseen	Harmonic
2:8	96.7 ± 2.0	96.0 ± 1.1	96.4 ± 1.5	85.8 ± 0.9	79.7 ± 4.4	82.4 ± 2.1
3:7	98.2 ± 0.7	74.4 ± 3.4	84.6 ± 2.5	86.5 ± 0.3	62.2 ± 4.9	72.0 ± 3.4
4:6	95.5 ± 1.1	80.4 ± 1.3	87.3 ± 0.7	83.8 ± 2.6	68.1 ± 4.1	74.5 ± 1.6
5:5	94.1 ± 2.5	55.4 ± 0.4	69.7 ± 0.9	84.7 ± 2.2	38.0 ± 3.0	51.5 ± 2.2
6:4	95.4 ± 0.1	33.2 ± 0.7	49.3 ± 0.7	83.7 ± 0.4	18.1 ± 2.9	29.1 ± 3.7
7:3	77.9 ± 0.1	22.3 ± 0.2	34.7 ± 0.3	88.1 ± 2.5	5.8 ± 0.9	10.8 ± 1.6