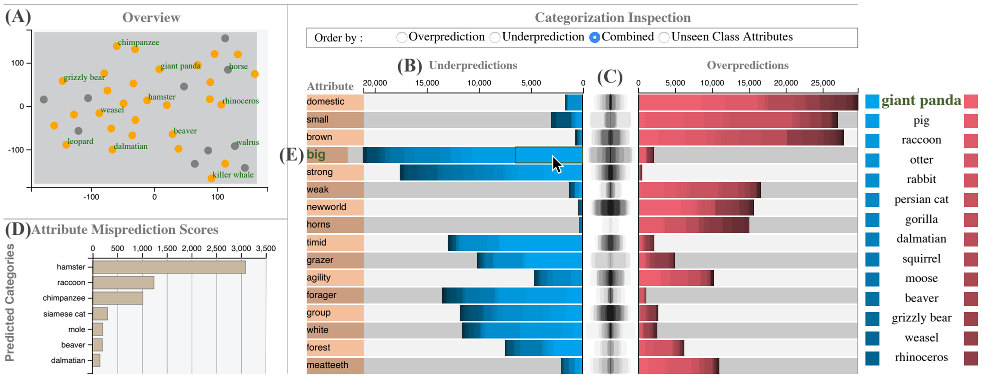
Visually Analyzing and Steering Zero Shot Learning

Saroj Sahoo[*]
Vanderbilt University

Matthew Berger[†]
Vanderbilt University



Order by : ○Overprediction ○Underprediction ●Combined ○Unseen Class Attributes

Figure 1: Our visual interface for diagnosing and steering the procedure of zero-shot learning. A scatterplot overview (A) enables the user to select categories for more detailed inspection. The main view (B-C) encodes category predictions with respect to attributes – human-interpretable properties that describe categories. For a given category and attribute, the user can inspect details (D) on category mispredictions made by the model. Based on these views, the user can modify attribute importance (E) to steer the model and mitigate errors when applied to data associated with categories not seen during training.

Abstract

We propose a visual analytics system to help a user analyze and steer zero-shot learning models. Zero-shot learning has emerged as a viable scenario for categorizing data that consists of no labeled examples, and thus a promising approach to minimize data annotation from humans. However, it is challenging to understand where zero-shot learning fails, the cause of such failures, and how a user can modify the model to prevent such failures. Our visualization system is designed to help users diagnose and understand mispredictions in such models, so that they may gain insight on the behavior of a model when applied to data associated with categories not seen during training. Through usage scenarios, we highlight how our system can help a user improve performance in zero-shot learning.

Index Terms: Zero-shot learning—Visualization—Visual Analytics—Model Steering

1 Introduction

Many of the recent successes in machine learning are owed to the ample amounts of supervised data provided by humans. Yet humans cannot simply be treated as an unlimited resource of supervision, as it is costly, in terms of both time and cognitive load, for humans to annotate data. Hence, the problem of zero-shot learning (ZSL) [29] has emerged as one way to support scenarios in machine learning where human supervision is scarce. Specifically, the typical setup in ZSL is to build a model on a dataset where categories are provided (*seen*), in order to apply the model to data associated with categories not provided during the model's construction (*unseen*). To solve this problem, ZSL models typically rely on *attributes* – semantic and human-nameable properties that characterize categories – to establish a relationship between categories associated with data seen during training, and unseen categories. However, this transfer between seen and unseen categories faces several challenges. Some attributes might not be particularly informative for the learning task at hand, while relevant attributes might be difficult to accurately model. A deeper understanding of attributes can help support the analytic tasks of both consumers and developers of ZSL methods, in terms of analyzing where and why mispredictions occur, and how to edit the model to mitigate error when applied to unseen categories.

In this work, we propose a visual analytics system to analyze and steer ZSL models. Our approach is attribute-centric: we diagnose mispredictions in terms of attributes to convey potential failure modes of ZSL. Although significant work has been recently developed in the visual analytics community for diagnosing and understanding the performance of machine learning models [9, 18], ZSL presents a unique challenge. Specifically, in the visualization, we do not have *any* access to data associated with categories for which the model will ultimately be used. Adhering to the traditional ZSL setup, our visualization only has access to data associated with seen categories. On the other hand, we also assume access to category-level attributes, both for seen and unseen categories. The main goal of our work is to use this provided information to help the user – be it model developer or consumer – make good decisions on how to *steer* the model, through analyzing, identifying, and modifying attributes in terms of their *reliability* for categorization.

The main contributions of our work can be summarized as follows: (1) We show how to extract information from predictions made by a ZSL model that help explain model errors. (2) We summarize this information in a visualization design that supports the identification of errors in seen categories, and how such errors might transfer to data associated with unseen categories. (3) Our design supports model steering, and we show how a modification of ZSL can support such user feedback. (4) We show through usage scenarios how our system can support a user in understanding a ZSL model for image categorization, and how to improve its performance.

2 Related Work

Our approach is related to methods in visual analytics for understanding, diagnosing, and steering machine learning models; please

[*] e-mail: saroj.k.sahoo@vanderbilt.edu
[†] e-mail: matthew.berger@vanderbilt.edu

see Hohman et al. [7] for an overview on the types of questions users have when interacting with models. Within model understanding, approaches based on *local interpretability* help explain why a prediction was made through locally-built model proxies [19, 20], using a learned rule list to explain predictions [15], diagnosing why an error occurred in a prediction [3, 9], or potential changes to features that impact a prediction [11]. On the other hand, *global interpretability* focuses on summarizing a set of predictions. Prior work has considered how to present the confidence of classifiers in multi-class settings [18], grouping feature subsets into coherent predictions [9], and visually analyzing discriminative features [10, 30].

Visual analytics for model steering focuses on how to take approaches that aid in understanding models to inform the user on how to edit the model to improve performance. Approaches in model steering must identify information, from both the model and data, that is useful for users in editing a model [24]. Prior work has considered this for constructing decision trees [25], while other work has developed methods for building ensemble models through visualizing confusion matrices [23] and joint scatterplots of data and models [22]. Our steering scheme is inspired by work that learns distance functions via direct manipulation [2, 8].

Existing approaches for visually understanding and steering models, however, are not directly applicable to the case of ZSL, for several reasons. First, all such approaches assume that labeled data samples exist for all categories apriori. In ZSL, we need to determine what information from the training data, and resulting model, is useful to present to a user, without access to data of unseen categories. This is necessary for making effective decisions on adjusting the model, in order to improve model performance when eventually deployed. Secondly, techniques that aim to understand predictions via feature importance, either at a local [9, 19] or global [10] level, again due to lack of data for test categories. However, we assume access to attributes of unseen categories, and thus we utilize the attribute vectors of categories, in addition to the data mapping into the attribute space, for diagnosing errors in predictions.

3 Visual Analytics for Zero-Shot Learning

In this section we describe the problem of zero-shot learning (ZSL), and the tasks we aim to address in our visualization design. ZSL is applicable to numerous problem domains [17, 28], but to focus our work, we consider multi-class image categorization, which has received significant attention [29]. In this setting, the goal is to train a model on images of *seen* categories, that can recognize images associated with categories *unseen* at training. The transfer of knowledge from seen to unseen categories is done with the help of attributes – a set of human-nameable properties that describes a category. For instance, if our categories consist of animals, then attributes are properties of animals, e.g. *furry*, *paws*, *fast*, that are shared between animals, e.g. bobcat and siamese cat both have **paws**, but the latter is *faster* than the former. Attributes enable a common space between seen and unseen categories that can be used for categorization: given an image, we predict its attributes, and then find the category whose attributes are most similar to the image's attributes. Given this setting, a major goal of ZSL is the characterization of unseen categories as *combinations of attributes* from seen categories. For instance, suppose category leopard was unseen, but bobcat and siamese cat were seen. All three categories have many attributes in common, e.g. paws, quadrupedal, that would allow the model to distinguish an image of a bobcat from a different, unseen category, e.g. a whale.

However, the performance of ZSL is highly dependent on its ability to accurately model attributes. In the above example, it is possible for a model to underpredict the attribute *fast* for unseen category leopard, and thus confuse it as a different, but related, category e.g. persian cat. These issues are well-understood in the community [14, 16]. Namely, ZSL methods can suffer from the *hubness* problem, where an image's predicted attributes become a hub for category attributes, reducing the model's discriminative power, while there is also a tendency to *bias* the image's attributes towards the seen categories and thus reduce the effectiveness in mapping to unseen categories. Yet, there remains a lack of methods for more detailed inspection, that can help diagnose problems in ZSL. In particular, we would like to gain insight on where a model is likely to *overpredict* or *underpredict* certain attributes. This can be useful to understand how such a model might behave on unseen categories, and in particular, what can be done to modify the model to prevent potential errors.

These issues inform the goals we aim to address:

G1 – Understand a ZSL model on seen categories, namely incorrect classifications, and the relationship to attribute predictions.

G2 – Depict model behavior for unseen categories, e.g. how the model might make poor attribute predictions.

G3 – Enable model steering: prioritize attributes for prediction in unseen categories.

We address these goals by supporting the following tasks:

T1 – Provide category overviews, and category selection, for more detailed, downstream analysis of categories [G1,G2].

T2 – Enable attribute-centric exploration to highlight errors in seen categories [G1].

T3 – Compare attribute-based errors in seen categories with unseen categories [G2].

T4 – Support user editing of attribute importance to mitigate prediction error in unseen categories [G3].

4 Visualization Design

In this section we discuss the ZSL model we use, the data collected from the model, as well as the design of the visualization.

4.1 ZSL Model and Data

Our visualization is designed to support ZSL models that learn to transform the input data into an attribute space. To this end, our model is based on Akata et al. [1]. This approach can be broken down into two main components: learning a mapping from input to attribute space, and optimizing a max-margin loss driven by an attribute-based compatibility function.

Mapping to Attribute Space. In this step, our goal is to define a function f that maps the d-dimensional input $\mathbf{x}$ to the a-dimensional attribute space. We adopt Akata et al. [1] and represent f as a 2-layer neural network, though our design could support other, related, approaches [1, 13, 21, 31].

Optimizing a Compatibility Function. Given the mapping, in this step our goal is to optimize a categorization criterion. This requires a way to define a compatibility between an input data instance, and a category – both represented as attributes. The compatibility function, s, we use is the dot product between points in the attribute space, specifically : $s(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\intercal \mathbf{z}_j$, where $\mathbf{z}_i$ and $\mathbf{z}_j$ are a-dimensional attribute vectors. Combined with the attribute mapping function, this permits us to measure the similarity between data $\mathbf{x}$ and an attribute vector for a category $\mathbf{z}$ via $s(f(\mathbf{x}), \mathbf{z})$. Given s, the loss we use to optimize for f maximizes the margin [1] of compatibilities between an input's ground truth category denoted y_i, and all other seen categories denoted $y \in S$:

$$\underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} \max_{y \neq y_i} \lfloor s(f(\mathbf{x}_i), \mathbf{z}_y) - s(f(\mathbf{x}_i), \mathbf{z}_{y_i}) + \eta \rfloor_+, \tag{1}$$

where η is a margin hyperparameter and $\lfloor x \rfloor_+$ returns x if the expression is positive, and zero otherwise. Once optimized, we may use both the learned mapping f and the compatibility function s to categorize a data input $\mathbf{x}$ from a set of unseen categories, denoted U, via $\operatorname{argmax}_{y \in U} s(f(\mathbf{x}), \mathbf{z}_y)$.

In order to visually diagnose prediction errors that are made by the model, we first take a closer look at the max-margin loss for a

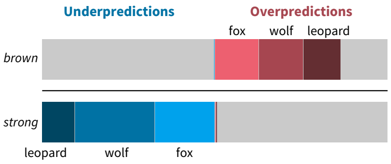


Figure 2: We visually encode underpredictions and overpredictions of attributes via a diverging and stacked bar plot, where underpredictions of categories are stacked to the left, and overpredictions to the right, and each row encodes an attribute.

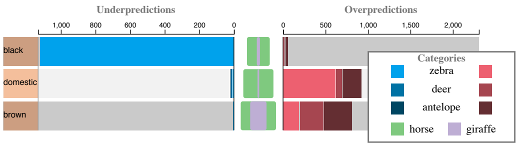


Figure 3: Our design for encoding unseen category attributes, alongside attribute errors in seen category, enables visual analysis between seen and unseen categories.

single data instance $\mathbf{x}_i$ (discarding the margin, as it does not affect predictions):

$$\max_{y \neq y_i} s(f(\mathbf{x}_i), \mathbf{z}_y) - s(f(\mathbf{x}_i), \mathbf{z}_{y_i}) = \max_{y \neq y_i} f(\mathbf{x}_i)^{\intercal}(\mathbf{z}_y - \mathbf{z}_{y_i}). \tag{2}$$

Assume that the model predicts some category y that renders the expression positive, indicative of a misprediction. As we are using a dot product-based similarity, the above expression can be rewritten as:

$$\sum_{k=1}^{a} p_y^k(\mathbf{x}_i), \tag{3}$$

where the summand is $p_y^k(\mathbf{x}_i) = f_k(\mathbf{x}_i)(\mathbf{z}_{y,k} - \mathbf{z}_{y_i,k})$ for each attribute k. Whenever $p_y^k(\mathbf{x}_i) > 0$, this indicates the mapping f is poor – we are either *overpredicting* or *underpredicting* this attribute for the given data $\mathbf{x}_i$, relative to its incorrect prediction y_i. Specifically, we are overpredicting if $f(\mathbf{x}_i) > 0$, and underpredicting if $f(\mathbf{x}_i) < 0$. It is precisely these summands we would like the user to inspect in our visual interface, to understand mispredictions, e.g. what attributes are being overpredicted/underpredicted, and by how much (**T2**).

4.2 Category Overview

We first present to the user an overview with respect to all categories (**T1**). Specifically, we perform t-SNE [26] with respect to the category attribute vectors, obtaining a 2D scatterplot of categories – both seen and unseen. We use t-SNE to ensure that local structure is retained in the projection, namely, categories with similar attributes will be close to one another. In the scatter plot we visually encode the seen categories with the color *blue* whereas we encode unseen categories with color *red*. We enable selection of seen categories via rectangular brushing, and unseen categories via clicking on individual points, where upon selection, seen categories are visually encoded as *orange* and unseen categories as *grey*, please see Fig. 1(A).

4.3 Visually Exploring Attributes of Seen Categories

Our main view summarizes mispredictions made by the model, captured by the scores $p_y^k(\mathbf{x})$. This view is prioritized by attributes, so that the user can explore incorrect predictions due to attributes with respect to user-selected categories. In particular, we would like to distinguish overpredictions from underpredictions. To this end, for a given attribute k and category y, we sum up all scores for data whose categories are y, individually for over/underpredictions:

$$q_{k,y}^{+/-} = \sum_{(\mathbf{x}_i, y_i) \in D_s} p_y^k(\mathbf{x}) \begin{cases} \text{if} \quad y_i = y \;,\; f(\mathbf{x}_i) > 0 \\ \text{if} \quad y_i = y \;,\; f(\mathbf{x}_i) < 0 \end{cases} , \tag{4}$$

namely q^+ corresponds to overpredictions $f(\mathbf{x}_i) > 0$ and q^- corresponds to underpredictions $f(\mathbf{x}_i) < 0$. The above assumes equality in the number of data instances per category – in practice we introduce a per-category scale to account for imbalance. We visually encode q^+ and q^- through a stacked and diverging bar plot, where each attribute is mapped to a row, q^+ is mapped to the right side of the baseline, and q^- is mapped to the left, please see Fig. 2 for an illustration. For a given attribute, we stack bars based on user-selected categories from the scatterplot. The stacking order is determined based on the sum of the over/under predictions over all attributes, such that categories with a higher sum are positioned closer to the baseline. We also encode this order via a sequential colormap, for a fixed hue of red and blue for over and under predictions, respectively. This view is central in our design, to support precise comparisons between categories and attributes. For example, in Fig. 2 explainable mispredictions can be quickly determined (**T2**), e.g. leopards, wolves, and foxes are poorly categorized because the model underpredicts the attribute *strong*, and overpredicts the attribute *brown*.

We populate an additional view that decomposes the data used to compute $q_{k,y}^{+}$ or $q_{k,y}^{-}$ in terms of their false positives (**T2**), shown in Fig. 1(D). Upon hovering over a bar, this view is populated such that mispredicted categories are mapped to rows, and each bar encodes the summation factors in q specific to its category.

4.4 Analyzing Unseen Categories

Only analyzing seen categories tells us little about the model's behavior on unseen categories. We only have access to attribute vectors for unseen categories, and not specific data instances. Yet our visualization is designed around attributes, and thus, we visually encode the relationship between attribute vectors of unseen categories, with attribute errors made on seen categories (**T3**).

To combine the attribute vectors of unseen categories with the main view, we length-encode the attributes by placing them in the center of the diverging stacked bar chart. Based on the selection made by the user we support 2 different analysis scenarios:

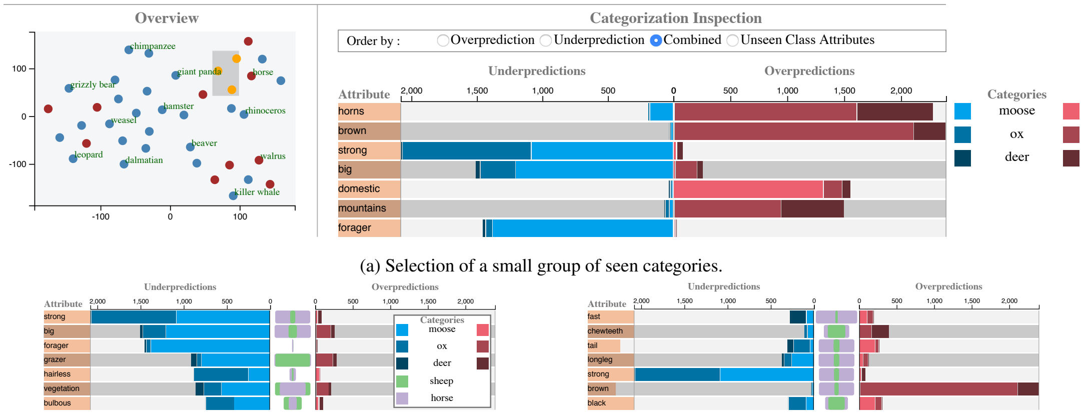
Detailed Category Analysis. If only a single unseen category is selected, each attribute of this category is encoded with a bar in the center, allowing us to relate unseen category attributes with seen category errors. If exactly 2 unseen categories are selected, we enable the user to compare unseen categories, where a category with lower attribute is layered on top of the category with higher attribute – please see Fig. 3 for such a case.

Unseen Category Overview. If a user selects more than 2 categories, we treat this as a case of analyzing category overviews. We use an ordinal color map, where we map the total number of categories that overlap in a particular attribute to a grey-scale value, shown in Fig. 1(C) highlights this case.

Additionally, to help the user find visually salient patterns between error in seen category predictions and the unseen category attributes, inspired by LineUp [5] we allow the user to sort by the under/over prediction scores, the sum of the over and under prediction scores, and the sum of attribute values for all selected unseen categories. The different ordering schemes allow the user to prioritize their analysis.

4.5 Model Steering

As part of the user's exploration, we support model steering by decreasing the importance of individual attributes, and retraining



(a) Selection of a small group of seen categories.

(b) Selecting two unseen classes and ordering by underpredictions.

(c) Ordering by unseen class attributes.

Figure 4: We show a scenario for locally exploring categories. First, the user selects seen categories close together in the scatterplot (a). The user then selects nearby unseen categories, ordering attributes by underpredictions (b). Sorting by summed unseen category values (c), the user can find attributes of different values for unseen categories, and subsequently relate unseen and seen categories in terms of misprediction scores.

the model from these weights. Specifically, when a user observes problematic attributes, they may click on any bar of that attribute in the main view, and this will decrease the importance of that attribute (**T4**). Fig. 1 highlights the results of this process, where in (E) we visually encode the weights by the length of a layered orange-colored bar. Initially all weights are set to 1, and each time the user clicks on a bar we decrease the corresponding weight by 0.1.

To retrain the model we form a diagonal matrix $D \in \mathbb{R}^{a \times a}$, where D_{ii} contains the user's weight for attribute i. We then modify our compatibility function by incorporating this matrix, leading to:

$$s_D(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\mathsf{T} D \mathbf{z}_j. \tag{5}$$

We then substitute s with s_D in Eq. 1, and optimize the resulting loss. The model thus gives less importance to attributes found problematic by the user, and to focus on more reliable attributes.

5 Case Studies

We present a use case of our system, to show how one would analyze the potential behavior of unseen categories. In our experiments we use the Animals with Attributes dataset [12, 29]. This dataset is comprised of 50 animal-based categories and 85 attributes, where we use ImageNet-trained ResNet features [6] of the images as our data input, provided by Xian et al. [29]. On average, there are approximately 750 images per category in the dataset. Please refer to the supplementary material for further details regarding the model, training, and testing

Step 1. The user first selects a subset of categories in the scatterplot to obtain an overview of attribute mispredictions, c.f. Fig. 4a. They focus their attention on a small cluster of categories – moose, ox, and deer – all of which share semantically-similar attributes.

Step 2. The user then finds unseen categories in the scatterplot that are close to the seen categories, and clicks on sheep and horse. The unseen category attribute view is then populated in the center of the baseline, where the user first orders the attributes by underpredictions, shown in Fig. 4b. Here we can observe that sheep are weaker and smaller than horses, and for these attributes we observe high underpredictions over related seen categories, thus there is the potential to incorrectly classify a horse as a sheep by underpredicting *strong* and *big*. Motivated by this observation, the user then prefers to see other attributes where horses and sheep differ, and thus reorders by the sum of the unseen attribute values as shown in Fig. 4c. Within this new ordering, the user discovers a set of attributes that have similar characteristics regarding horse and sheep. In particular, they find the attribute *brown* is highly overpredicted, and that horses are generally considered more brown than sheep. Thus, the model is likely to mispredict images of sheep as horses on the basis of attribute *brown*. Furthermore, the *tail* attribute is shown to be equally likely to overpredict or underpredict, indicating a rather unreliable attribute, which could potentially impact the categorization of unseen classes.

Step 3. After identifying unreliable attributes like *brown* and *tail* the user next steers the model by decreasing their weights, as shown in Fig. 4c. By retraining the model, we find that the user is able to improve the accuracy of predicting category sheep from 47.6% to 81.1%, while also improving the overall performance of the model from 53.2% to 55.1%.

In general, we find this localized analysis applies quite well to other unseen categories. We have performed similar model steering experiments centered on other categories, and found encouraging results – please see the supplemental material for further results. One potential downside to local analysis is that an improvement in one category might result in a decreased accuracy for another category, leading to potential bias. Global analysis of unseen categories – as depicted in Fig. 1 – can help counter such bias, thus a mixture of these two analyses is most likely to be useful in practice.

6 Discussion

We think that our approach is an important step in using visual analytics to help understand zero-shot learning, and there are several directions we would like to explore as part of future work. We have, thus far, only considered our approach for the Animals with Attributes dataset, and so we intend to use our interface for other ZSL datasets. Though our approach should scale well to datasets such as CUB [27], which is comprised of 200 categories and 312 attributes, larger datasets such as ImageNet, comprised of 1,000 categories, will necessitate different visual encodings and interactions. Our diverging, stacked bar design is most useful for comparing tens of categories across several attributes at a time, thus alternative designs, ones that carefully aggregate both category and attribute-based information, will be developed to support the analysis of larger-scale datasets. We also found it challenging to properly decrease attribute weights through our model steering, as setting attribute weights to be too small can adversely impact performance, though we generally found positive results by decreasing weights to the range of [0.5,0.7]. The problem of translating human judgement of model errors to model weights [4], however, is quite challenging, and we believe outside of the scope of our work.

References

[1] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2927–2936, 2015.

[2] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 83–92. IEEE, 2012.

[3] N.-C. Chen, J. Suh, J. Verwey, G. Ramos, S. Drucker, and P. Simard. Anchorviz: Facilitating classifier error discovery through interactive semantic data exploration. In *23rd International Conference on Intelligent User Interfaces*, pp. 269–280. ACM, 2018.

[4] O. Evans, A. Stuhlmüller, C. Cundy, R. Carey, Z. Kenton, T. McGrath, and A. Schreiber. Predicting human deliberative judgments with machine learning. Technical report, Technical report, University of Oxford, 2018.

[5] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. Lineup: Visual analysis of multi-attribute rankings. *IEEE transactions on visualization and computer graphics*, 19(12):2277–2286, 2013.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[7] F. Hohman, A. Head, R. Caruana, R. DeLine, and S. M. Drucker. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, p. 579. ACM, 2019.

[8] X. Hu, L. Bradel, D. Maiti, L. House, and C. North. Semantics of directly manipulating spatializations. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2052–2059, 2013.

[9] J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs, and E. Bertini. A workflow for visual diagnostics of binary classifiers using instance-level explanations. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 162–172. IEEE, 2017.

[10] J. Krause, A. Perer, and E. Bertini. Infuse: interactive feature selection for predictive modeling of high dimensional data. *IEEE transactions on visualization and computer graphics*, 20(12):1614–1623, 2014.

[11] J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 5686–5697. ACM, 2016.

[12] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 951–958. IEEE, 2009.

[13] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2013.

[14] A. Lazaridou, G. Dinu, and M. Baroni. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 270–280, 2015.

[15] Y. Ming, H. Qu, and E. Bertini. Rulematrix: visualizing and understanding classifiers with rules. *IEEE transactions on visualization and computer graphics*, 25(1):342–352, 2018.

[16] A. Paul, N. C. Krishnan, and P. Munjal. Semantically aligned bias reducing zero shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7056–7065, 2019.

[17] R. Qiao, L. Liu, C. Shen, and A. Van Den Hengel. Less is more: zero-shot learning from online textual documents with noise suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2249–2257, 2016.

[18] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics*, 23(1):61–70, 2016.

[19] M. T. Ribeiro, S. Singh, and C. 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*, pp. 1135–1144. ACM, 2016.

[20] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pp. 2152–2161, 2015.

[22] B. Schneider, D. Jäckle, F. Stoffel, A. Diehl, J. Fuchs, and D. Keim. Visual integration of data and model space in ensemble learning. In *2017 IEEE Visualization in Data Science (VDS)*, pp. 15–22, 2017.

[23] J. Talbot, B. Lee, A. Kapoor, and D. S. Tan. Ensemblematrix: interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1283–1292. ACM, 2009.

[24] G. K. Tam, V. Kothari, and M. Chen. An analysis of machine-and human-analytics in classification. *IEEE transactions on visualization and computer graphics*, 23(1):71–80, 2016.

[25] S. Van Den Elzen and J. J. van Wijk. Baobabview: Interactive construction and analysis of decision trees. In *2011 IEEE conference on visual analytics science and technology (VAST)*, pp. 151–160. IEEE, 2011.

[26] L. Van Der Maaten. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pp. 384–391, 2009.

[27] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[28] S. Wu, S. Bondugula, F. Luisier, X. Zhuang, and P. Natarajan. Zero-shot event detection using multi-modal fusion of weakly supervised concepts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2665–2672, 2014.

[29] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[30] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics*, 25(1):364–373, 2018.

[31] Z. Zhang and V. Saligrama. Zero-shot learning via semantic similarity embedding. In *Proceedings of the IEEE international conference on computer vision*, pp. 4166–4174, 2015.