Evaluating Systemic Error Detection Methods using Synthetic Images

Gregory Plumb [*1] **Nari Johnson** [*1] **Ángel Alexander Cabrera** [1] **Marco Tulio Ribeiro** [2] **Ameet Talwalkar** [1]

Abstract

We introduce SpotCheck, a framework for generating synthetic datasets to use for evaluating methods for discovering blindspots (*i.e.,* systemic errors) in image classifiers. We use SpotCheck to run controlled studies of how various factors influence the performance of blindspot discovery methods. Our experiments reveal several shortcomings of existing methods, such as relatively poor performance in settings with multiple blindspots and sensitivity to hyperparameters. Further, we find that a method based on dimensionality reduction, PlaneSpot, is competitive with existing methods, which has promising implications for the development of interactive tools.

1. Introduction

A growing body of research has found that machine learning models with high test performance often make systemic errors (Buolamwini and Gebru, 2018; Chung et al., 2019; Oakden-Rayner et al., 2020; Ribeiro et al., 2020; Singla et al., 2021; Ribeiro and Lundberg, 2022), which occur when the model performs significantly worse on a coherent (*i.e.,* semantically meaningful) subset of the data. For example, past works (Winkler et al., 2019; Mahmood et al., 2021) have demonstrated that models trained to diagnose skin cancer from dermoscopic images rely on spurious artifacts (such as the presence of a surgical skin marker that some dermatologists use to mark lesions) to make predictions. As a result, these models have different performance on images with or without those spurious artifacts. More broadly, discovering systemic errors is critical in a range of applications, such as detecting algorithmic bias (Buolamwini and Gebru, 2018) or sensitivity to distribution shifts (Sagawa et al., 2020; Singh et al., 2020).

[*]Equal contribution [1]Carnegie Mellon University [2]Microsoft Research. Correspondence to: Gregory Plumb <gdplumb@andrew.cmu.edu>, Nari Johnson <narij@andrew.cmu.edu>.

Published at the ICML 2022 Workshop on Spurious Correlations, Invariance, and Stability. Baltimore, Maryland, USA. Copyright 2022 by the author(s).

While there is a rich body of work that studies how to find systemic errors in settings where there is useful metadata to define coherent subsets (Buolamwini and Gebru, 2018; Chung et al., 2019; Cabrera et al., 2019; Singh et al., 2020), finding systemic errors is much harder in settings without such metadata. For example, we often do not know a-priori which images may or may not have some artifact (such as a surgical skin marker). To address this challenge, several Blindspot Discovery Methods (BDMs)[1], such as Sohoni et al. (2020); Singla et al. (2021); d'Eon et al. (2021); Eyuboglu et al. (2022), have been proposed to discover blindspots in settings without useful metadata.

While the motivation and technical approaches of these BDMs are well defined, there is no standardized approach for evaluating them. Evaluating the hypothesized blindspots returned by BDMs is fundamentally challenging because it is unclear how to measure their coherence and because we do not have the complete set of true blindspots to compare them against. One approach to address these challenges is to compare the hypothesized blindspots to a subset of the true blindspots that have either been artificially induced or identified by existing work (Sohoni et al., 2020; Eyuboglu et al., 2022). While this is a promising direction, it makes it difficult to measure a BDM's recall or false positive rate or to identify various factors that influence a BDM's performance.

To address these challenges, we propose a synthetic evaluation framework for BDMs called SpotCheck. SpotCheck builds on ideas from Kim et al. (2022) by generating synthetic datasets with varying degrees of complexity and then training models on those datasets to have various types of blindspots. This allows us to measure a BDM's recall and false positive rate, since we know the full set of true blindspots for each model, and to measure how various factors, such as the number or types of blindspots in the model, influence a BDM's performance.

We use SpotCheck to conduct an evaluation of 3 recent BDMs and compare them to a new baseline method called PlaneSpot. Our evaluation reveals several insights about BDM performance: performance degrades quickly as the number of blindspots in a model increases, performance

[1]Past works have used other terms such as "discovering failure modes," "unknown systemic errors," or "slices." We chose "blindspot" because we believe it is more descriptive and concise.

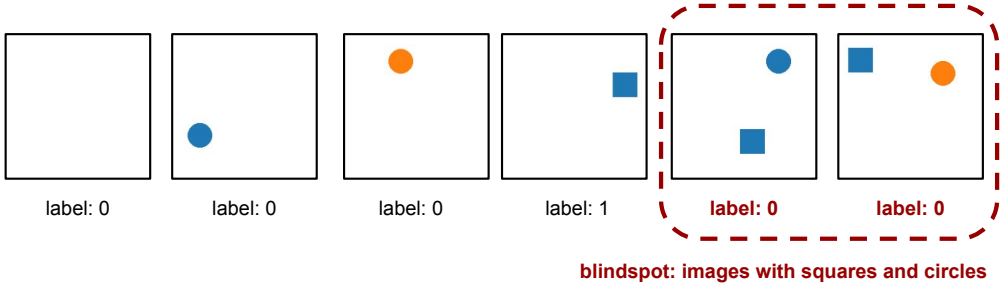

Figure 1: A simple example created with `SpotCheck`. **Dataset Complexity.** This dataset is defined by 3 semantic features that vary across images: the presence of a square, the presence of a circle, and the color of the circle. We do not count the "color of the square" because it is always blue. **Blindspot Specificity.** This blindspot is defined by 2 semantic features: the presence of a square and the presence of a circle. As a result, it contains any image with both a square and a circle, regardless of the circle's color. **Training Labels.** In general, the label for each image indicates if a square is present. However, any training or validation image belonging to this blindspot is mislabeled.

depends on the type of features that define a blindspot, and performance is sensitive to the BDM's hyperparameters.

Interestingly, we observe that `PlaneSpot` consistently outperforms existing BDMs. Because `PlaneSpot` utilizes a 2D representation, our findings suggest that it could be useful in an interactive setting. We plan to explore this direction using real datasets in future work.[2]

2. Synthetic Evaluation for BDMs

`SpotCheck` builds on ideas from Kim et al. (2022) to generate synthetic datasets of varying complexity and to train models with specific blindspots on those datasets. Below, we summarize its key steps; see Appendix A for details.

Dataset Definition. Each dataset can be defined using *semantic features* that describe the possible types of images it contains. Datasets that have a larger number of features have a larger variety of images and are therefore more complex. For example, a simple dataset may only contain images with squares and blue or orange circles (see Figure 1), while a more complicated dataset may also contain images with striped rectangles, small text, or grey backgrounds.

Blindspot Definition. Each blindspot is defined using a subset of the semantic features that define its associated dataset (see Figure 1). Similarly to how a dataset with more features is more complex, a blindspot defined using more semantic features is more specific.

Training a Model with Specific Blindspots. For each dataset and blindspot specification, we train a ResNet-18 model (He et al., 2016) to predict whether a square is present.

[2]All code will be released at https://github.com/user/repo

To induce blindspots, we generate data where the label for each image in the training and validation sets is correct if and only if it does not belong to any of the blindspots (see Figure 1). The test set images are always correctly labeled.

Generating Diverse Experimental Configurations. Since our goal is to study how various factors influence BDM performance, we generate a diverse set of *experimental configurations*, (*i.e.,*dataset, blindspots, and model triplets). To do this, we randomize the features that define a dataset (both the number of them and what they are) as well as the blindspots (the number of them, the number of features that define them and what those features are).

3. Evaluation Metrics

Each BDM returns an *ordered list* of hypothesized blindspots, $\hat{\mathbf{\Psi}} : [\hat{\Psi}_k]_{k=1}^{K}$, sorted by decreasing importance. Further, because we are using `SpotCheck`, we have the complete set of the model's true blindspots, $\mathbf{\Psi} : \{\Psi_m\}_{m=1}^{M}$. Our goal is to measure how well the hypothesized $\hat{\mathbf{\Psi}}$ capture the true $\mathbf{\Psi}$, where each $\hat{\Psi}_k$ and Ψ_m are sets of images. We start by measuring how well a BDM finds each *individual* true blindspot (Blindspot Recall) and build on that to measure how well a BDM finds the *complete set* of true blindspots (Discovery Rate and False Discovery Rate).

Blindspot Precision. We start by checking if $\hat{\Psi}_k$ is a subset of Ψ_m. If it is, we know that the model underperforms on $\hat{\Psi}_k$ and that $\hat{\Psi}_k$ is coherent. We measure this using the precision of $\hat{\Psi}_k$ with respect to Ψ_m:

$$\text{BP}(\hat{\Psi}_k, \Psi_m) = \frac{|\hat{\Psi}_k \cap \Psi_m|}{|\hat{\Psi}_k|} \tag{1}$$

We say that $\hat{\Psi}_k$ *belongs to* Ψ_m if, for some threshold λ_p:

$$\text{BP}(\hat{\Psi}_k, \Psi_m) > \lambda_p \tag{2}$$

However, $\hat{\Psi}_k$ can belong to Ψ_m without capturing the same information as Ψ_m. For example, $\hat{\Psi}_k$ could be "squares and blue circles" while Ψ_m could be "squares and blue or orange circles". Because this excessive specificity could result in the user arriving at insufficiently general conclusions, we need to incorporate some notion of recall into the evaluation.

Blindspot Recall. One approach to measure recall is to calculate the proportion of Ψ_m that $\hat{\Psi}_k$ covers *individually*:

$$\text{BR}_{\text{naive}}(\hat{\Psi}_k, \Psi_m) = \frac{|\hat{\Psi}_k \cap \Psi_m|}{|\Psi_m|} \tag{3}$$

We relax this definition by allowing Ψ_m to be covered by the union of *multiple* of the $\hat{\Psi}_k$ that belong to it:

$$\text{BR}(\hat{\mathbf{\Psi}}, \Psi_m) = \frac{\left| \left(\bigcup_{\hat{\Psi}_k : \text{BP}(\hat{\Psi}_k, \Psi_m) > \lambda_p} \hat{\Psi}_k \right) \cap \Psi_m \right|}{|\Psi_m|} \tag{4}$$

We say that $\hat{\mathbf{\Psi}}$ *covers* Ψ_m if, for some threshold λ_r:

$$\text{BR}(\hat{\mathbf{\Psi}}, \Psi_m) > \lambda_r \tag{5}$$

We do this because "squares and blue circles" and "squares and orange circles" belong to and jointly cover "squares and blue or orange circles." So, if a BDM returns both, a user could combine them to arrive at the correct conclusion.

Discovery Rate (DR). We define the *discovery rate* of $\hat{\mathbf{\Psi}}$ and $\mathbf{\Psi}$ as the fraction of the Ψ_m that are covered by $\hat{\mathbf{\Psi}}$:

$$\text{DR}(\hat{\mathbf{\Psi}}, \mathbf{\Psi}) = \frac{1}{M} \sum_m \mathbb{1}(\text{BR}(\hat{\mathbf{\Psi}}, \Psi_m) > \lambda_r) \tag{6}$$

False Discovery Rate (FDR). When the DR is non-zero, we define *false discovery rate* of $\hat{\mathbf{\Psi}}$ and $\mathbf{\Psi}$ as the fraction of the $\hat{\Psi}_k$ that do not belong to any of the $\mathbf{\Psi}$:[3]

$$\text{FDR}(\hat{\mathbf{\Psi}}, \mathbf{\Psi}) = \frac{1}{K} \sum_k \mathbb{1}(\max_m BP(\hat{\Psi}_k, \Psi_m) \leq \lambda_p) \tag{7}$$

Note that, without the complete set of true blindspots (as in `SpotCheck`) it is impossible to calculate FDR.

[3]While calculating DR, we may only need the top-u items of $\hat{\mathbf{\Psi}}$. As a result, we only calculate the FDR over those top-u items. This prevents the FDR from being overly pessimistic when we intentionally pick K too large in our experiments. However, when the DR is zero, it is not clear what value of u to use, so we exclude the FDR from our analysis.

Method	DR	FDR
Barlow	0.43 (0.04)	0.03 (0.01)
Spotlight	0.79 (0.03)	0.09 (0.01)
Domino	0.64 (0.04)	0.07 (0.01)
`PlaneSpot`	**0.85 (0.03)**	**0.03 (0.01)**

Table 1: Average BDM DR and FDR along with their standard errors across 100 experimental configurations.

4. `PlaneSpot`

`PlaneSpot` starts from the representation defined by the model's penultimate layer and uses scvis (Ding et al., 2018) to learn a 2D embedding of that representation. This 2D embedding is then normalized, so each dimension has range [0, 1], and a 3rd dimension, $w *$ ModelConfidence(x, y), is appended. This 3D input is passed to a Gaussian Mixture Model clustering algorithm where the number of clusters is chosen using the Bayesian Information Criterion. The clusters are then sorted by the product of their error rate and the number of errors in them. w is a hyperparameter that controls the relative importance of the 2D embedding of an image and the model's confidence for that image.

5. Experiments

We use `SpotCheck` to generate 100 experimental configurations whose datasets have 6-8 semantic features and whose models have 1-3 blindspots with 5-7 features. We evaluate the following BDMs: Spotlight (d'Eon et al., 2021), Barlow (Singla et al., 2021), and Domino (Eyuboglu et al., 2022). For each BDM, we use a held-out set of 20 configurations to select hyperparameters. We use $\lambda_p = \lambda_r = 0.8$.

Overall Results. Table 1 shows the DR and FDR results averaged across all 100 experimental configurations. We observe that, in comparison to other methods, `PlaneSpot` has the highest DR, on average finding 85% of the true blindspots per experimental configuration. `PlaneSpot` and Barlow have a lower FDR than Spotlight and Domino.

Identifying factors that influence BDM performance. We study two types of factors: *holistic factors*, which measure properties of the dataset (*e.g.,* how complex is it?) or of the model (*e.g.,* how many blindspots does it have?), and *specific factors*, which measure properties of a blindspot (*e.g.,* is it defined with this feature?). For holistic factors, we average DR and FDR across the experimental configurations. For specific factors, we find the "fraction of true blindspots covered" averaged across each individual blindspot from the experimental configurations (see Equation 5).

The number of blindspots matters. In Figure 2, we plot the average DR for experimental configurations with $1, 2,$

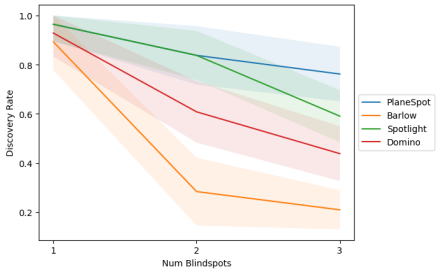

Figure 2: Average BDM DR (and 95% confidence intervals indicated by the shaded regions) for experimental configurations that have 1, 2, and 3 true blindspots.

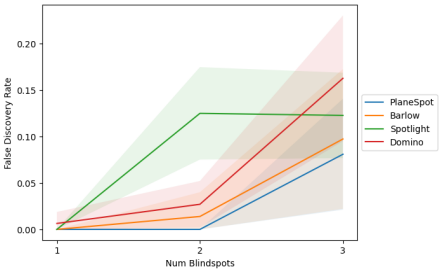

Figure 3: Average BDM FDR for experimental configurations that have 1, 2, and 3 true blindspots.

and 3 true blindspots. Average DR decreases for all methods as the number of blindspots increases. Figure 3, shows that FDR increases as this happens as well. The conclusion that methods perform worse in settings with multiple blindspots is particularly significant because past evaluations have primarily focused on settings with one blindspot.

The specificity of blindspots matters. In Figure 4, we plot the fraction of true blindspots covered for blindspots defined using $5, 6$, and 7 features. With the exception of Spotlight, all of these methods are less capable of finding more specific/less frequently occurring blindspots.

The features that define a blindspot matter. In Figure 5, we plot the fraction of true blindspots covered for blindspots that are or are not defined using the "relative position" feature; this feature is an indicator for whether the square is above the image's horizontal center line. With the exception of `PlaneSpot`, all methods are less likely to find blindspots defined using this feature. This shows that the types of features used to define a blindspot (*e.g.,* the presence of spurious objects, color or texture information, background information) can influence BDM performance.

There should be more discussion on hyperparameter tuning. In Figure 6, we observe that two hyperparameter settings that perform nearly identically on average exhibit significantly different performance at identifying blindspots

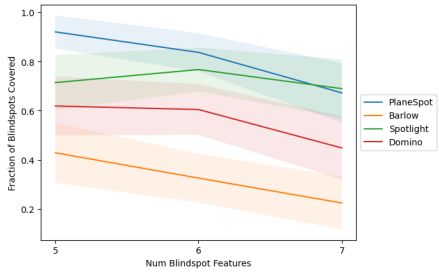

Figure 4: The fraction of true blindspots covered, averaged over the individual blindspots from the experimental configurations, for blindspots defined using $5, 6$, and 7 features.

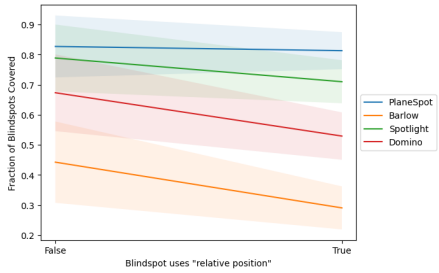

Figure 5: The fraction of true blindspots covered that are or are not defined with the "relative position" feature.

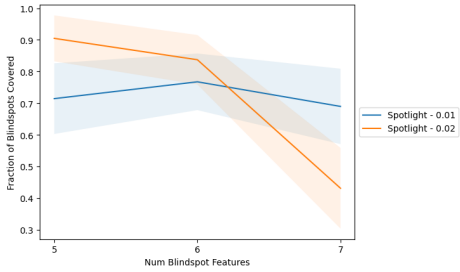

Figure 6: Despite the fact that these hyperparameter choices perform similarly on average, they behave differently based on the number of features used to define the blindspot.

defined using differing numbers of features. This suggests that there may not be a single best hyperparameter choice to discover all of the blindspots in a single model, which could contain multiple blindspots of different specificity or frequency. In conjunction with the general sensitivity that these methods have to their hyperparameters, this suggests that there should be more discussion on hyperparameter tuning (especially since this is much easier in our controlled setting than in real applications).

6. Related Work

Finding Blindspots. Numerous methods have been proposed to help users discover blindspots across a wide set of applications. We focus on methods that make the least restrictive assumptions and work for image classification models. Specifically, we evaluate the methods from Singla

Method	Image Representation	Dimensionality Reduction	Hypothesis Class
Multiaccuracy (Kim et al., 2019)	VAE embedding		Linear model
GEORGE (Sohoni et al., 2020)	Model embedding	UMAP ($d = 0, 1, 2$)	Gaussian kernels
Spotlight (d'Eon et al., 2021)	Model embedding		Gaussian kernels
Barlow (Singla et al., 2021)	Adversarially-Robust Model embedding		Decision Tree
Domino (Eyuboglu et al., 2022)	CLIP embedding	PCA ($d = 128$)	Gaussian Kernels
`PlaneSpot`	Model embedding	scvis ($d = 2$)	Gaussian Kernels

Table 2: A high level overview the major design choices made by different BDMs.

et al. (2021); d'Eon et al. (2021); Eyuboglu et al. (2022) because they do not assume any of the following:

- Access to metadata (Kim et al., 2018; Buolamwini and Gebru, 2018; Chung et al., 2019; Singh et al., 2020; Plumb et al., 2021).
- Access to tools that manipulate data (Shetty et al., 2019; Singla et al., 2020; Xiao et al., 2021; Leclerc et al., 2021; Bharadhwaj et al., 2021)
- Any specific model structure (Alvarez-Melis and Jaakkola, 2018; Koh et al., 2020).
- Any specific model training process (Higgins et al., 2017; Tsipras et al., 2019; Wong et al., 2021)
- A human in the loop, either through an interactive interface (Cabrera et al., 2019; Balayn et al., 2022; Ribeiro and Lundberg, 2022) or inspecting explanations (Yeh et al., 2020; Adebayo et al., 2021).

Table 2 summarizes the design choices of these methods:

- They use a model to extract a representation of an image. Typically, this is the same model whose blindspots we are trying to discover, but it can be a different model.
- They apply some form of dimensionality reduction to that image representation.
- They learn a model from a specified hypothesis class to predict if an image belongs to a blindspot from the image's (potentially reduced) representation. Note that this leaves out important details on how that model is learned.

We note that `PlaneSpot`'s main change is learning a 2D representation using scvis (Ding et al., 2018). Beyond this, it uses the most common choices for the 'Image Representation' and 'Hypothesis class' while also using standard techniques for learning a model from that hypothesis class.

Quantitative Evaluations of BDMs. One evaluation approach focuses on measuring properties of the hypothesized blindspots, such as their error rate or size (Singla et al., 2021; d'Eon et al., 2021). However, while these properties are important, they do not capture whether or not the hypothesized blindspots are coherent.

Another approach is to compare the hypothesized blindspots to a subset of the true blindspots that have been artificially induced or identified in prior work (Sohoni et al., 2020; Eyuboglu et al., 2022). This approach is similar to `SpotCheck`, with a few key differences:

- Past work uses unrealistic definitions of what it means to discover a true blindspot that either only considers precision (Eyuboglu et al., 2022) or considers both precision and recall but uses thresholds for precision that are too lenient (Sohoni et al., 2020). Further, neither allow hypothesized blindspots to be combined.
- Without access to the complete set of true blindspots, they cannot measure method DR or FDR.
- They do not isolate factors that influence a BDM's performance (*e.g.,* number of blindspots, the specificity of those blindspots, or the features that define the blindspots).

7. Conclusion

We propose `SpotCheck`, a synthetic evaluation framework for BDMs, and ran controlled studies of how various factors influence BDM performance. This evaluation yields fundamental insights about when and why different BDMs are less effective and is an important step towards formalizing a more rigorous and complete set of desiderata for BDMs. However, it remains a question for follow-up work to see if our observed trends generalize to settings with real images; in general, we believe that poor performance on synthetic data implies poor performance on real data, but not necessarily the other way around.

Our experimental results have many interesting implications for future work. First, it is important to evaluate BDMs in settings with models that have multiple blindspots, as this setting is more realistic and challenging. Second, finding a way to tune BDM hyperparameters in realistic settings is an open challenge with significant practical importance.

Finally, we are intrigued by the result that `PlaneSpot`, which learns a GMM on a 2D embedding, performs competitively with prior methods, which use significantly higher-dimensional embeddings. In future work, we hope to evaluate `PlaneSpot` on real image data and to explore methods to visualize its 2D embedding as the basis for an interactive blindspot discovery tool.

References

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1550–1553. IEEE, 2019.

Luke Oakden-Rayner, Jared Dunnmon, Gustavo Carneiro, and Christopher Ré. Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. In *Proceedings of the ACM conference on health, inference, and learning*, pages 151–159, 2020.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, 2020.

Sahil Singla, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. Understanding failures of deep networks via robust feature extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12853–12862, 2021.

Marco Tulio Ribeiro and Scott Lundberg. Adaptive testing and debugging of nlp models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3253–3267, 2022.

Julia K Winkler, Christine Fink, Ferdinand Toberer, Alexander Enk, Teresa Deinlein, Rainer Hofmann-Wellenhof, Luc Thomas, Aimilios Lallas, Andreas Blum, Wilhelm Stolz, et al. Association between surgical skin markings in dermoscopic images and diagnostic performance of a deep learning convolutional neural network for melanoma recognition. *JAMA dermatology*, 155(10):1135–1141, 2019.

Usman Mahmood, Robik Shrestha, David DB Bates, Lorenzo Mannelli, Giuseppe Corrias, Yusuf Emre Erdi, and Christopher Kanan. Detecting spurious correlations with sanity tests for artificial intelligence guided radiology systems. *Frontiers in digital health*, page 85, 2021.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxGuJrFvS.

Krishna Kumar Singh, Dhruv Mahajan, Kristen Grauman, Yong Jae Lee, Matt Feiszli, and Deepti Ghadiyaram. Don't judge an object by its context: Learning to overcome contextual bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11070–11078, 2020.

Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. Fairvis: Visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 46–56. IEEE, 2019.

Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Advances in Neural Information Processing Systems*, 33:19339–19352, 2020.

Greg d'Eon, Jason d'Eon, James R Wright, and Kevin Leyton-Brown. The spotlight: A general method for discovering systematic errors in deep learning models. *arXiv preprint arXiv:2107.00758*, 2021.

Sabri Eyuboglu, Maya Varma, Khaled Kamal Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Re. Domino: Discovering systematic errors with cross-modal embeddings. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=FPCMqjI0jXN.

Joon Sik Kim, Gregory Plumb, and Ameet Talwalkar. Sanity simulations for saliency methods. *ICML*, 2022.

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.

Jiarui Ding, Anne Condon, and Sohrab P Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature communications*, 9 (1):1–13, 2018.

Michael P Kim, Amirata Ghorbani, and James Zou. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 247–254, 2019.

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

Gregory Plumb, Marco Tulio Ribeiro, and Ameet Talwalkar. Finding and fixing spurious patterns with explanations. *arXiv preprint arXiv:2106.02112*, 2021.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. Not using the car to see the sidewalk – quantifying and controlling the effects of context in classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1xFWgrFPS.

Kai Yuanqing Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gl3D-xY7wLq.

Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. *arXiv preprint arXiv:2106.03805*, 2021.

Homanga Bharadhwaj, De-An Huang, Chaowei Xiao, Anima Anandkumar, and Animesh Garg. Auditing ai models for verified deployment under semantic specifications. *arXiv preprint arXiv:2109.12456*, 2021.

David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7786–7795, 2018.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SyxAb30cY7.

Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11205–11216. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wong21b.html.

Agathe Balayn, Natasa Rikalo, Christoph Lofi, Jie Yang, and Alessandro Bozzon. How can explainability methods be used to support bug identification in computer vision models? In *CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2022.

Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020.

Julius Adebayo, Michael Muelly, Harold Abelson, and Been Kim. Post hoc explanations may be ineffective for detecting unknown spurious correlation. In *Proceedings of the 2022 International Conference on Learning Representations*, 2021.

A. Generating Experimental Configurations, Extended

In this section we detail how we use `SpotCheck` to generate random experimental configurations.

- In Section A.1, we define the different types of semantic features that can appear in each image.
- In Section A.2, we define a synthetic image dataset, how we generate random datasets, and how we sample images from a dataset.
- In Section A.3, we define a blindspot for a synthetic image dataset, how we generate a random blindspot, and how we generate an unambiguous set of blindspots.

A.1. Semantic Features

Table 3 defines all of the semantic features that `SpotCheck` uses to generate synthetic images. We call these semantic features *Attributes* and group them into *Layers* based on what part of an image they describe. Each Attribute has two possible *Values*: a Default and Alternative Value. Each synthetic image has an associated list of (Layer, Attribute, Value) triplets that describes the image. Figure 7 shows this triplet list for two synthetic images.

We sometimes refer to the Square/Rectangle/Circle/Text Layers as *Object Layers* because they all describe a specific object that can be present in an image. The location of each object within an image is chosen randomly, subject to the constraint that each object doesn't overlap with any other object.

Table 3: The Layers and Attributes that define the synthetic images.

Layer	Attribute	Default Value	Alternative Value
Background	Color	White	Grey
	Texture	Solid	Salt and Pepper Noise
Square/Rectangle/Circle/Text	Presence	False	True
	Size	Normal	Small
	Color	Blue	Orange
	Texture	Solid	Vertical Stripes
Square (continued)	Number	1	2

A.2. Defining a Dataset using these Semantic Features

At a high level, `SpotCheck` defines a *Dataset* by deciding whether or not each Attribute of each Layer is *Rollable* (*i.e.,* the Attribute can take either its Default or Alternative Value, uniformly at random) or not Rollable (*i.e.,* the Attribute only takes its Default Value). We measure a Dataset's complexity using the number of Rollable Attributes it has. Figure 7 describes the Rollable and Not Rollable Attributes for an example Dataset.

Generating a Random Dataset. We start by picking which Layers will be part of the Dataset:

- Images need a background, so all Datasets have the Background Layer.
- The task is to predict whether there is a square in the image, so all Datasets have the Square Layer.
- We add 1-3 (chosen uniformly at random) of the other Object Layers (chosen uniformly at random without replacement from the set {Rectangle, Circle, Text}) to the Dataset.

Once the Layers are chosen, we make 6-8 (chosen uniformly at random) of the Attributes Rollable:

- Each Object Layer has its Presence Attribute made Rollable.
- Then, the remaining Rollable Attributes are chosen by iteratively:
  - Selecting a Layer uniformly at random from those that have at least one Not Rollable Attribute.
  - Selecting an Attribute from that Layer uniformly at random from those that are Not Rollable.

Sampling an Image from a Dataset. Once a Dataset's Rollable Attributes have been defined, generating a random image is straightforward:

- For each Attribute from each Layer in the Dataset, we pick a random Value if the Attribute is Rollable. Attributes that are Not Rollable will take their Default Value.
  - If the Layer is an Object Layer:
    - If the Presence Attribute is True, the location of the object is chosen randomly (subject to the non-overlapping constraint).
    - If the Presence Attribute is False, the object will not be rendered (regardless of the Values chosen for the other Attributes of this Layer).
- We then use the resulting (Layer, Attribute, Value) triplet list and the list of object locations to render a 224x224 RGB image.
- Finally, we calculate any MetaAttributes (explained next) and append these (Layer, MetaAttribute, Value) triplets to the image's definition list.

Calculating MetaAttributes. While each Attribute corresponds to a semantic feature, there are a potentially infinite number of MetaAttributes that one could calculate as semantically meaningful functions of an image. We list the MetaAttributes that we calculate in our experiments in Table 4. Because this space is infinitely large and grows with the number of Attributes, we exclude MetaAttributes from our measure of Dataset complexity.

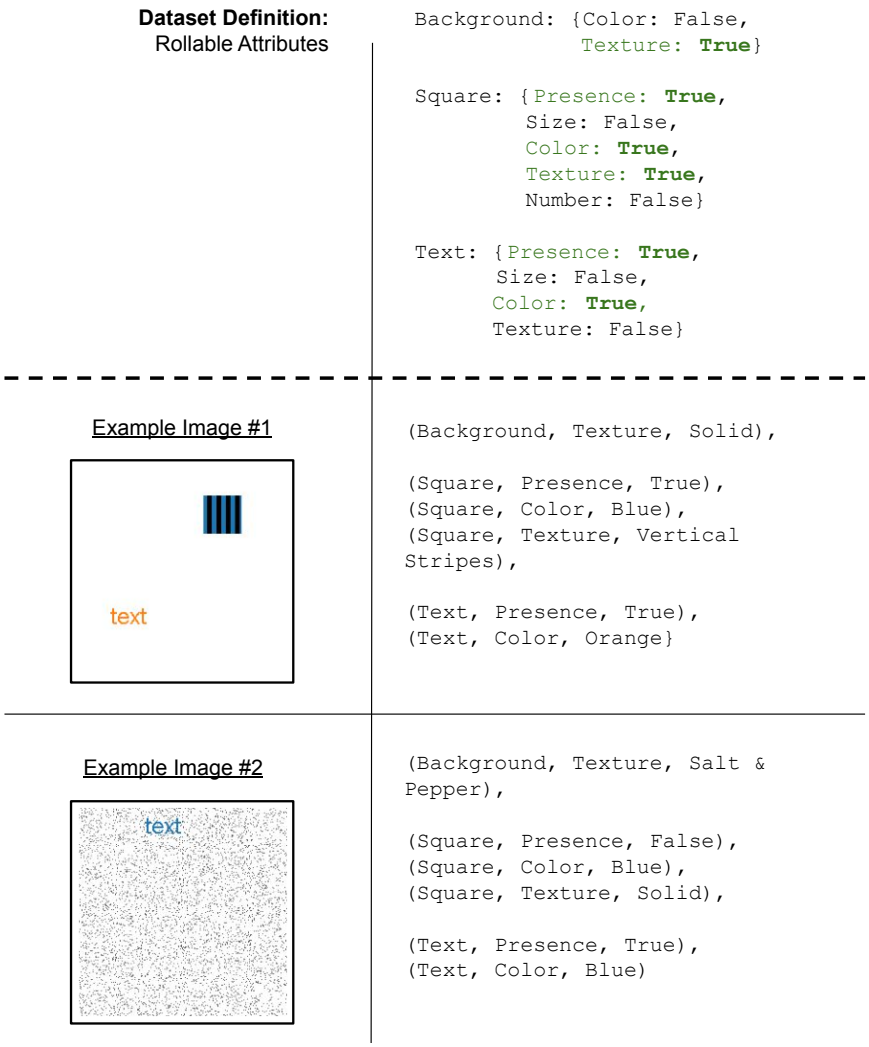

Figure 7: **Top Row.** The definition of an example Dataset generated by `SpotCheck`. Notice that this Dataset has 3 Layers and 6 Rollable Attributes. **Middle/Bottom Row.** Two example images generated from this Dataset along with their (Layer, Attribute, Value) triplet lists. Notice that Not Rollable Attributes in this Dataset take on their Default Values in these images and are not in the images' triplet lists.

Table 4: The MetaAttributes that we calculate for each synthetic image.

Layer	MetaAttribute	Value	Meaning
Background	Relative Position	1	Square is above the horizontal centerline of the image
		0	Square is bellow the horizontal centerline of the image
		-1	No Square

A.3. Defining the Blindspots for a Dataset

`SpotCheck` defines a *Blindspot* using a list of (Layer, (Meta)Attribute, Value) triplets. We measure a Blindspot's specificity using the length of its definition list. An image belongs to a blindspot if and only if the Blindspot's definition list is a subset of the image's definition list. Figure 8 shows two example Blindspots.

Generating a Random Blindspot. `SpotCheck` generates a random Blindspot consisting of 5-7 (chosen uniformly at random) (Layer, (Meta)Attribute, Value) triplets for a Dataset by iteratively:

- Selecting a Layer (uniformly at random from those that have at least one Rollable Attribute[4] that is not already in this Blindspot)
- Selecting a Rollable Attribute from that Layer:
  - Object Layers: If the Layer's Presence Attribute is not in this Blindspot, select its Presence Attribute. Otherwise, select an Attribute uniformly at random from those that are not already in this Blindspot and set the Layer's Presence Attribute Value to True for this Blindspot.
  - Background Layers: Select an Attribute uniformly at random from those that are not already in this Blindspot.
- Selecting a Value for that Attribute (uniformly at random)

Notice that, if an Object Layer is selected more than once, then we ensure that the Object's Presence Attribute has a Value of True in the Blindspot definition. We enforce this *Feasibility Constraint* to ensure that every triplet in the Blindspot's definition list correctly describes the images belonging to the Blindspot (*e.g.,* [(Circle, Presence, False), (Circle, Color, Blue)] is infeasible because an image with a blue circle must have a circle in it).

Generating an Unambiguous Set of Blindspots. For each Dataset, we generate 1-3 (chosen uniformly at random) Blindspots using the process described above. However, when generating multiple blindspots, they can be *ambiguous* which causes problems when using them to evaluate BDMs.

Definition. A set of Blindspots, S_1, is ambiguous if there exists a different set of Blindspots, S_2, such that both:

1. The union of images belonging to S_1 is equivalent to the union of images belonging to S_2. As a result, S_1 and S_2 would both correctly describe the model's blindspots.
2. An evaluation that uses Discovery Rate (Equation 6) would penalize a BDM if it returns S_2 instead of S_1. More precisely, $\text{DR}(S_2, S_1) < 1$ for $\lambda_p = \lambda_r = 1$.

Example. Suppose that we have a very simple Dataset with two Rollable Attributes, X and Y which are uniformly distributed and independent, and consider two different sets of Blindspots for this Dataset:

- $S_1 = \{B_1, B_2\}$ where $B_1 = [(X = 1)]$ and $B_2 = [(X = 0), (Y = 1)]$
- $S_2 = \{B_1^{'}, B_2^{'}\}$ where $B_1^{'} = [(X = 1), (Y = 0)]$ and $B_2^{'} = [(Y = 1)]$

Then, S_1 is ambiguous because:

- S_1 and S_2 induce the same behavior in the model: they both mislabel an image if $X = 1 \vee Y = 1$.
- A BDM would be penalized for returning S_2:

$$\text{BP}(B_1^{'}, B_1) = 1.0 \wedge \text{BP}(B_1^{'}, B_2) = 0 \wedge \text{BP}(B_2^{'}, B_1) = \text{BP}(B_2^{'}, B_2) = 0.5 \implies$$

$$\text{BR}(S_2, B_1) = 0.5 \wedge \text{BR}(S_2, B_2) = 0 \implies$$

$$\text{DR}(S_2, S_1) = 0$$

In fact, for this example, there are only two sets of two unambiguous Blindspots, ($\{[(X = 0), (Y = 0)], [(X = 1), (Y = 1)]\}$ and $\{[(X = 0), (Y = 1)], [(X = 1), (Y = 0)]\}$), and there exists no unambiguous set of three Blindspots.

Preventing Ambiguity. In general, ambiguity occurs whenever the union of two blindspots forms a contiguous region in the discrete space defined by the Rollable Attributes. Consequently, we prevent ambiguity by ensuring that any pair of blindspots has at least two of the *same* Rollable Attributes with *different* Values in their definition lists. We call this the *Ambiguity Constraint*.

Implications of the Ambiguity and Feasibility Constraints. In our experiments, our goal is to generate experimental

[4] All MetaAttributes are considered to be "Rollable" when generating a random Blindspot.

configurations with a diverse set of Datasets and associated Blindspots. However, the Ambiguity Constraint (AC) and Feasibility Constraint (FC) limit the number of valid Blindspots for any specific Dataset.

To see this, notice that the AC places more constraints on each successive Blindspot added to an experimental configuration. This has two implications. First, that generating an experimental configuration with more Blindspots requires a Dataset with more Rollable Attributes (more complexity) and Blindspots with more triplets (more specificity). Further, because we cannot set the Attribute Values of a Blindspot's triplets independently of each other [FC], we need more complexity and specificty than a simple analysis based only on the AC suggests. Second, that each successive Blindspot is more closely related to the previous ones which means that larger sets of Blindspots are "less diverse" or "less random" in some sense.

With these trade-offs in mind, we generated experimental configurations with:

- Background, Square, and 1-3 other Object Layers
- A total of 6-8 Rollable Attributes
- 1-3 Blindspots
- 5-7 triplets per Blindspot

because an experimental configuration with any combination of these values is able to satisfy the AC and the FC while still having a diverse set of Blindspots.

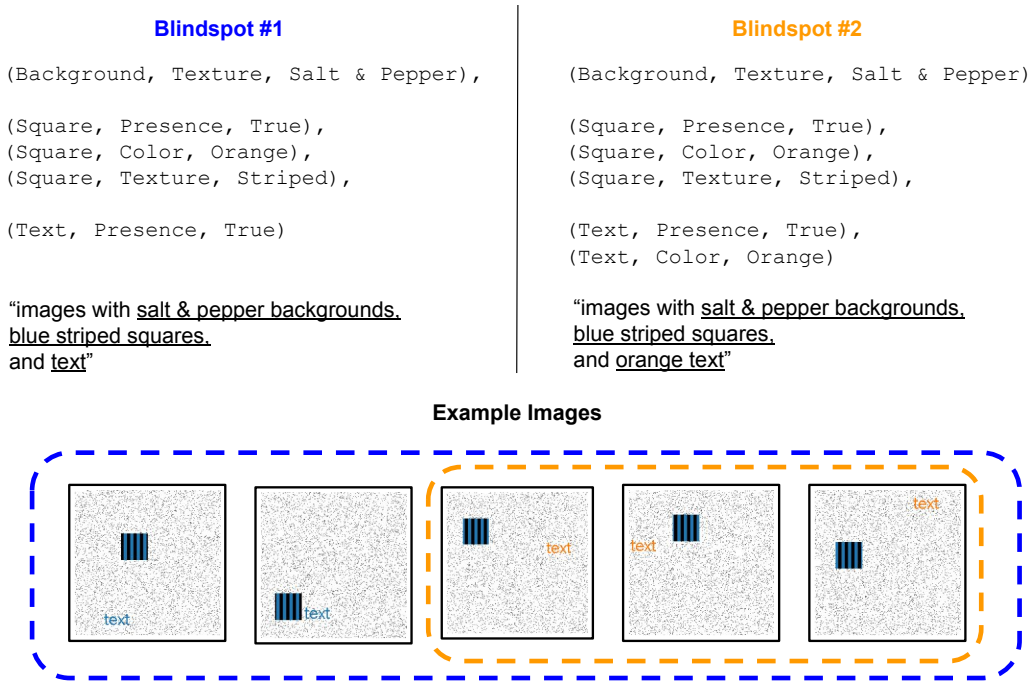

Figure 8: Two example Blindspots, Blindspot #1 (Left) and Blindspot # 2 (Right) for the example Dataset from Figure 7. Each Blindspot is defined by a list of (Layer, (Meta)Attribute, Value) triplets as shown above. We also display example images belonging to each Blindspot: all of the images inside of the blue border belong to Blindspot #1, while only the subset of images inside of the orange border belong to Blindspot #2. In this example, Blindspot #2 is more specific (defined using 6 semantic features) than Blindspot #1 (defined using 5 semantic features).