

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 5 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 8 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 5: 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 8 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 6. Because this space is infinitely large and grows with the number of Attributes, we exclude MetaAttributes from our measure of Dataset complexity.

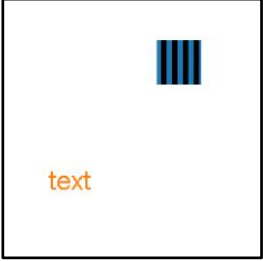
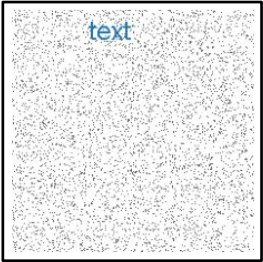
<p>Dataset Definition: Rollable Attributes</p>	<pre>Background: {Color: False, Texture: True} Square: {Presence: True, Size: False, Color: True, Texture: True, Number: False} Text: {Presence: True, Size: False, Color: True, Texture: False}</pre>
<p><u>Example Image #1</u></p> 	<pre>(Background, Texture, Solid), (Square, Presence, True), (Square, Color, Blue), (Square, Texture, Vertical Stripes), (Text, Presence, True), (Text, Color, Orange)</pre>
<p><u>Example Image #2</u></p> 	<pre>(Background, Texture, Salt & Pepper), (Square, Presence, False), (Square, Color, Blue), (Square, Texture, Solid), (Text, Presence, True), (Text, Color, Blue)</pre>

Figure 8: **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 6: 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 9 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 Attribute³ 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, $DR(S_2, S_1) < 1$ 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 :

$$BP(B'_1, B_1) = 1.0 \wedge BP(B'_1, B_2) = 0 \wedge BP(B'_2, B_1) = BP(B'_2, B_2) = 0.5 \implies$$

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

$$DR(S_2, S_1) = 0$$

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

³All MetaAttributes are considered to be “Rollable” when generating a random Blindspot.

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 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 specificity 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.

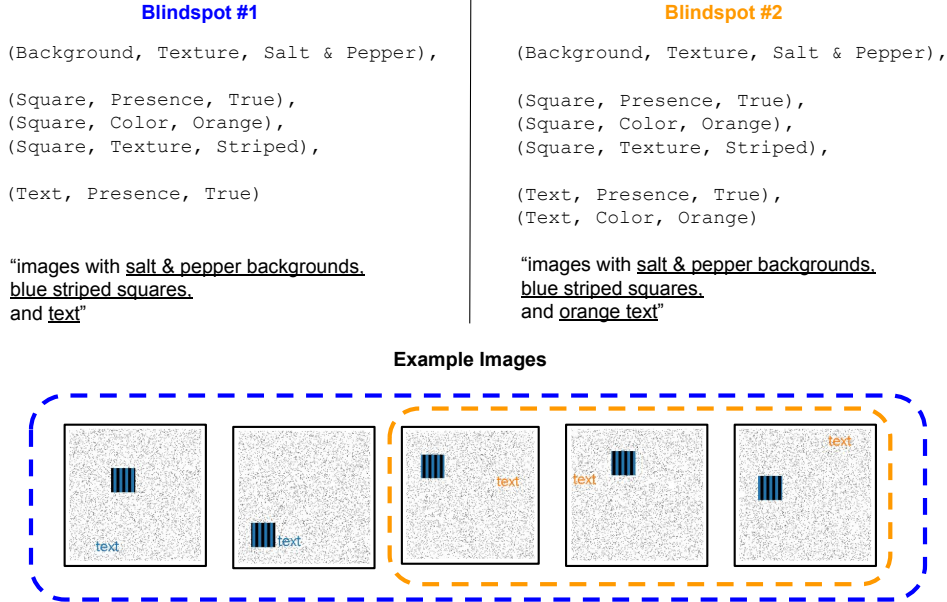


Figure 9: Two example Blindspots, Blindspot #1 (Left) and Blindspot #2 (Right) for the example Dataset from Figure 8. Each Blindspot is defined by a list of (Layer, (Meta)Attribute, Value) triplets as shown above. We also display example images belonging to each Blindspot: all of the images inside of 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).

B HOW DO OUR DEFINITIONS OF “BELONGING TO” AND “COVERING” BUILD ON SIMILAR IDEAS IN EXISTING WORK?

At a high level, we refine these ideas so that they better reflect the fact that, in practice, a user is going to look at the hypothesized blindspots returned by a BDM in order to come to conclusions about the model’s true blindspots. This entails making three specific changes.

First, our definition of “belonging to” uses a high, in absolute terms, threshold for λ_p (*i.e.*, 0.8) in order to minimize the amount of noise the user has to deal with when determining the semantic definition of a blindspot. This is in contrast to a relative definition (*e.g.*, better than random chance) which would lead to significant amounts of noise for uncommon blindspots. For example, Table 2 from Sohoni et al. (2020) shows that as few as one in seven images in the hypothesized blindspots are actually from the true blindspots for the Waterbirds and CelebA datasets. While better than random chance, the output of the BDM is still very noisy, which is likely to create problems for the user.

Second, our definition of “covering” incorporates blindspot recall in order to prevent the user from coming to conclusions that are too narrow. For example, consider the middle row of Figure 5 from Eyuboglu et al. (2022). In these examples, the blindspots are defined as “people wearing glasses,” “people with brown hair,” or “smiling people.” However, the BDM returns images that are of “men wearing glasses,” “women with brown hair, and “smiling women.” As a result, the user may incorrectly conclude that the model is exhibiting gender bias because the BDM led them to conclusions that are too narrow.

Third, our definition of “covering” allows for hypothesized blindspots to be combined in order to take advantage of the user’s ability to synthesize what they see and come to more general conclusions, which is something that would be very challenging to do algorithmically. Experimentally, we observe that this combining is essential for some of these BDMs to work on `SpotCheck`. Further, in Appendix F, we qualitatively observe that this is useful for real problems as well. For example, the “ties without people” blindspots includes a set of images of “cats wearing ties” that are separate, in the model’s representation space, from the other images in the blindspot and that, as a result, are likely to be returned separately by a BDM.

C HYPERPARAMETER TUNING

To tune hyperparameters, we use a secondary set of 20 ECs to calculate BDM DR and FDR. While some of these BDMs have multiple hyperparameters, we focused on the main hyperparameter for each BDM and left the others at their default values.

Barlow (Singla et al., 2021). The main hyperparameter is the *maximum depth* of the decision tree whose leaves define the hypothesized blindspots. The results of the hyperparameter search are in Table 7.

Spotlight (d’Eon et al., 2021). The main hyperparameter is the *minimum weight* assigned to a hypothesized blindspot. The results of the hyperparameter search are in Table 8.

Domino (Eyuboglu et al., 2022). The main hyperparameter is γ which controls the relative importance of coherence and under-performance in the hypothesized blindspots. The results of the hyperparameter search are in Table 9.

PlaneSpot. For our BDM, the main hyperparameter is w which controls the *weight* of the model-confidence dimension relative to the two spatial dimensions from the scvis embedding. The results of the hyperparameter search are in Table 10.

Table 7: The hyperparameter tuning results for Barlow. For maximum depths of 11 and 12, we noticed that Barlow’s outputs matched exactly for all of the configurations. This indicates that it never learned a decision tree of depth 12. As a result, we use a maximum depth of 11 for our experiments.

Maximum Depth	DR	FDR
3	0.07	0.0
4	0.14	0.0
5	0.19	0.0
6	0.27	0.12
7	0.36	0.15
8	0.37	0.09
9	0.43	0.11
10	0.43	0.08
11	0.48	0.07
12	0.48	0.07

Table 8: The hyperparameter tuning results for Spotlight. We use a minimum weight of 0.01 in our experiments. In Figure 2 we show that, despite the fact that 0.01 and 0.02 have very similar average results, they behave very differently.

Minimum Weight	DR	FDR
0.005	0.49	0.04
0.01	0.88	0.06
0.02	0.88	0.08
0.04	0.48	0.00

Table 9: The hyperparameter tuning results for Domino. We use $\gamma = 10$ in our experiments.

γ	DR	FDR
5	0.48	0.06
10	0.72	0.03
15	0.70	0.06
20	0.60	0.09

Table 10: The hyperparameter tuning results for PlaneSpot. We use $w = 0.025$ in our experiments.

w	DR	FDR
0	0.57	0.00
0.025	0.95	0.00
0.05	0.88	0.00
0.1	0.87	0.00

D WHY ARE THESE BDMs FAILING?

In this section, we are going to try to identify, from a methodological standpoint, why these BDMs are failing. We start by defining some additional metrics that allow us to understand why a BDM failed to cover a specific true blindspot. Then, we describe how we average those metrics across our ECs to observe general patterns. Finally, we give some possible explanations for the observed trends.

Explaining a Specific Failure. Given the output of a BDM, $\hat{\Psi}$, we want to understand why it failed to cover a specific true blindspot, Ψ_m . To do this, we measure the fraction of the images in that true blindspot, $i \in \Psi_m$, that fall into each of four categories:⁴

- i was *not returned* by the BDM. Intuitively, this tells us how often the BDM does not show an image to the user that it should. Specifically, this means that $i \notin \hat{\Psi}_k \forall k$.
- i is *found* by the BDM. Intuitively, this tells us how close the BDM is to covering Ψ_m . Specifically, this means that $\exists k \mid i \in \hat{\Psi}_k \wedge \text{BP}(\hat{\Psi}_k, \Psi_m) \geq \lambda_p$.
- i was part of a *merged* blindspot according to the BDM. Intuitively, this tells us how often the BDM merges multiple true blindspots into a single hypothesized blindspot. Specifically, this means that $\exists k \mid i \in \hat{\Psi}_k \wedge \text{BP}(\hat{\Psi}_k, \cup_m \Psi_m) \geq \lambda_p$. Note that $\text{BP}(\hat{\Psi}_k, \cup_m \Psi_m)$ measures the fraction of the images in $\hat{\Psi}_k$ that belong to *any* true blindspot.
- i was part of a *impure* blindspot according to the BDM. Intuitively, this tells us how often the BDM includes too many images that do not belong to any true blindspot into a hypothesized blindspot. Specifically, this means that $\text{BP}(\hat{\Psi}_k, \cup_m \Psi_m) < \lambda_p (\forall k \mid i \in \hat{\Psi}_k)$.

Averaging to gain more general patterns. While these metrics can help us understand why a BDM failed to cover a specific true blindspot from a specific EC, we want to arrive at more general results. To do this, we first average these metrics across the true blindspots in an EC that are not covered by the BDM, $\{\Psi_m \mid \text{BR}(\hat{\Psi}, \Psi_m) < \lambda_r\}$, and then average them across the ECs where the BDM did not cover all of the true blindspots, $\{\Psi \mid \text{DR}(\hat{\Psi}, \Psi) \neq 1\}$. Table 11 shows the results.

Explaining observed trends. Our main observation is that a significant portion of all of the BDMs’ failures can be explained by the fact that they tend to merge multiple true blindspots into a single hypothesized blindspot. However, this raises the question: is the a failure of the BDMs or are they simply inheriting the problem from their image representation, which is failing to separate the true blindspots? To address this question, we manually inspected the 2D scvis embedding used by PlaneSpot, which exhibits this failure the most strongly, for the 10 ECs where at least 50% of the images in the true blindspots that were not covered belonged to a hypothesized blindspot that merges multiple true blindspots. For 8 of those 10 ECs, the true blindspots were easily visually separable in this 2D embedding, which means that the true blindspots are also separable in the original image representation. Consequently, we conclude that this is a failing of the BDMs.

Additionally, we make two less significant observations. First, that Spotlight is the only BDM that fails to return a non-trivial fraction of the images that it is should. Because it is the only BDM that does not do this, we suggest that BDMs should partition the entire input space (including partitions that do not have higher than average error). Second, that Barlow is the only that has more failures due to returning impure hypothesized blindspots than merged blindspots. Between this and Barlow’s generally poor performance, this suggests that axis-aligned decision trees are not a particularly promising hypothesis class for BDMs.

Table 11: The average value of each of the four metrics we use to explain why a BDM failed to cover a true blindspot.

Method	Not Returned	Found	Merged	Impure
Barlow	0.02	0.15	0.39	0.44
Spotlight	0.13	0.46	0.33	0.04
Domino	0.0	0.38	0.36	0.24
PlaneSpot	0.0	0.0	0.57	0.42

⁴Every image, i , falls into exactly one of these categories. The definition of each category is written under the assumption that i does not fall into any of the previous categories.

E COCO EXPERIMENTS, DETAILS

We detail the methodology used to generate ECs using the COCO dataset (Lin et al., 2014).

E.1 PREDICTION TASK & DATA PREPROCESSING

COCO is a large-scale object detection dataset. We use the 2017 version of the dataset. We re-sample from the given train-test-validation splits to have a larger test set of images for use in blindspot discovery. Our final train, test, and validation sets have 66874, 44584, and 11829 images respectively.

Prediction Task. We define the binary classification task as detecting whether an object belonging to some *super-category* (containing multiple of the 80 object categories) is present in the image. In our experiments, each EC is associated with 1 of 3 different super-category prediction tasks: detecting if an *animal* (e.g., a dog, cat, etc), a *vehicle* (e.g., a car, airplane, etc), or a *furniture item* (e.g., a chair, bed, etc) are present in an image. We use the super-category definitions provided in the COCO paper (Lin et al., 2014). We sample ECs from multiple (as opposed to only a single) prediction tasks so that we have a larger number of possible blindspot definitions.

EC Preprocessing. For each EC, we use the EC’s prediction task to down-sample the original COCO dataset to ensure that the dataset’s blindspots are identifiable (*i.e.*, so that there is no ambiguity or overlap in the blindspots we induce in our models). Specifically, we drop all images that have more than 1 unique object category belonging to the task super-category (e.g., if the task is to detect animals, we drop all images that have both a dog and a cat). We also down-sample so that the final train, test, and validation sets are all class-balanced.

E.2 BLINDSPOT DEFINITIONS

All blindspots in our ECs are defined using an object category that is a subset of those belonging to the task super-category. For example, an EC with the “animal” prediction task can have the blindspot “zebra”, as the zebra object category belongs to the animal super-category (Figure 5).

Specificity. To measure the effect of blindspot specificity, we generate two different *types* of blindspots: more and less specific. Less-specific blindspots are defined using only one object category, such as the “zebra” blindspot. More-specific blindspots are defined using two object categories. Like the less-specific blindspots, one of the object categories must belong to the task super-category (e.g., must be an animal for an animal prediction EC). But, the second object category used to define a more-specific blindspot must *not* belong to the task super-category. For example, the second object can be “person” (but not “dog”) because a person is not an animal.

Each more-specific blindspot is then defined using (1) the presence of the object belonging to the super-category, and (2) either the presence *or absence* of the object outside of the super-category. For example, valid blindspot definitions include “images of elephants with people” (task: animal), “images of bicycles without a backpack” (task: vehicle), or “images of dining tables with forks” (task: furniture item). We chose to allow some blindspots to be defined using the *absence* of a second object inspired by the past observation that models can incorrectly rely on the presence of spurious artifact objects instead of identifying the true object class (Plumb et al., 2022), consequently underperforming on images without these artifacts.

E.3 MODEL TRAINING

Similarly to our experiments using SpotCheck, we induce true blindspots by training with the wrong labels for images belonging to blindspots. For each EC, we train a Resnet-18. We perform model selection by picking the model with the best validation set loss.

Validating that models learn the induced blindspots. Because real data is more complex than synthetic data, it is possible that even when we train with the wrong labels for a particular subgroup, the model may not underperform on other images belonging to the subgroup. We call this behavior *failing to induce* a true blindspot. For each true blindspot, we measure if the blindspot was successfully induced by comparing the model’s performance on images belonging to, vs. outside of the blindspot in the separate *test set*. We only retain ECs where the model has at least a 20% recall gap inside vs. outside of the blindspot.

E.4 ECs: GENERAL POOL

We began our analysis by replicating the procedure we used to sample random ECs using `SpotCheck` as closely as possible. Recall that when using `SpotCheck` to measure the effect of various factors, we generated a pool of ECs where each factor (*e.g.*, the number of model blindspots and each blindspot’s specificity) were chosen *independently*. We repeat this general procedure using the COCO dataset, and call these ECs the “General Pool”.

We provide pseudocode that we use to sample a pool of $N = 90$ ECs:

- For each super-category **prediction task** in {animal, vehicle, furniture item }:
 - ★ For each **number of blindspots** in [1, 2, 3]:
 - Generate 10 experimental configurations. For each configuration:
 - ★ For each blindspot, choose the blindspot’s definition:
 - Choose the blindspot’s **specificity** uniformly at random from {More-Specific, Less-Specific}.
 - Choose an **object category** that belongs to the task super-category uniformly at random from those that were not already used in the EC’s other blindspots.
 - If the blindspot is **more-specific**:
 - ★ Choose a second **object category** that does not belong to the task super-category uniformly at random from those that are *eligible*. An object category is eligible if it has not yet been used to define another of the ECs blindspots, and if both the intersection and set difference of the first and second object categories have at least 100 images in the train set. We impose these eligibility constraints to avoid selecting blindspots that are too small, as we hypothesize that larger blindspots may be easier to induce.
 - ★ Choose if the blindspot will be defined using the {Presence, Absence} of the second object uniformly at random.

Note that for each EC sampled using the above pseudocode, the EC’s number of blindspots and each individual blindspot’s specificity are selected independently.

Table 12: Counts the retained ECs in the general pool (where all blindspots are successfully induced) for each prediction task and number of blindspots.

Prediction Task	Number of blindspots	Count of (ECs retained for analysis) / (total ECs sampled)
animal	1	10 / 10
	2	8 / 10
	3	7 / 10
furniture	1	8 / 10
	2	6 / 10
	3	5 / 10
vehicle	1	8 / 10
	2	6 / 10
	3	7 / 10

Summary Statistics. Of the 90 sampled ECs in the general pool, we retain 65 for our analysis, dropping 25 where we failed to induce the sampled blindspot. Table 12 summarizes the prediction tasks and number of blindspots in the retained general pool of ECs.

E.5 ECs: CONDITIONED POOLS

Number of Blindspots: Conditioned Pool. We provide pseudocode that we use to sample a pool of $N = 84$ ECs. We hold each blindspot’s specificity constant and only sample less-specific blindspots.

Before sampling the conditioned pool ECs, we first train several models where for each model, we try to induce a single COCO object category as a blindspot. We sample the blindspots for the conditioned distribution only from the subset of C object categories that were successfully induced as singleton blindspots.

- For each super-category **prediction task** in {animal, vehicle, furniture item }:
 - ★ For each **number of blindspots** in [1, 2, 3]:
 - Using the C object categories that belong to the task super-category *and* that were able to be learned as singleton blindspots, sample up to $N = 20$ ECs from the list of C nCr (number of blindspots) possible combinations.

We only have 22 ECs with 1 (singleton) blindspot because there are only 22 total object categories that can be induced as singleton blindspots. All blindspots were successfully induced for the 84 sampled ECs. To measure the effect of the number of blindspots, we compare BDM performance the 22, 30, and 30 ECs with 1, 2, and 3 blindspots respectively.

Blindspot Specificity: Conditioned Pool. We provide pseudocode that we use to sample a pool of $N = 70$ ECs. We hold the number of blindspots constant and only sample ECs with a single blindspot.

- For each super-category **prediction task** in {animal, vehicle, furniture item }:
 - ★ Using the C object categories that belong to the task super-category *and* that were able to be learned as singleton blindspots, construct a list of all of the *eligible* blindspots:
 - For the less-specific blindspots, all C sub-categories are eligible.
 - For the more-specific blindspots, we define eligibility equivalently to the general pool: *i.e.*, for each C object categories belonging to the task super-category, a second object category (that does not belong to the task super-category) is a *eligible* if both the intersection and set difference of the first and second object categories have at least 100 images in the train set. Concretely, the list of eligible blindspots consists of all eligible pairs of objects with both options of whether the blindspot is defined using the {Presence, Absence} of the second object.
 - From the list of eligible blindspots, sample up to 20 ECs (each with 1 blindspot) without replacement.

Of the 60 generated more-specific ECs, we retain 48 ECs for analysis, dropping the 12 ECs where we failed to induce the sampled blindspot. Like the other conditioned pool, there are only 22 possible less-specific ECs. To measure the effect of blindspot specificity, we compare BDM performance on the 48 more-specific to the 24 less-specific ECs.

F OBSERVATIONS FROM A QUALITATIVE INSPECTION OF REAL DATA

In Section D, we used the 2D scvis embedding of the model’s learned representation as an analysis tool to determine that these BDMs’ tendency to merge true blindspots is a failing of the methods themselves. In this Section, we do something similar and qualitatively inspect that 2D embedding for models trained on real image datasets in order make additional observations that may be relevant to the design of future BDMs.

A low-dimensional embedding can work for real data. In the next subsections, we demonstrate that this 2D embedding contains the information required to identify a wide range of blindspots from prior work and to identify novel blindspots. As a result, it seems that dimensionality-reduction is a potentially important aspect of BDM design or that an interactive approach based on this embedding could be effective.

Getting a maximally general definition of a specific blindspot often requires merging multiple groups of images. For example, reaching the conclusions presented in Figures 15, 22, 27, and 37 all required merging several distinct groups of images. As a result, it seems that being able to suggest candidate merges is a potentially important goal for BDM design.

A small blindspot may appear as a small cluster (Figure 21) or as a set of outliers (Figure 33). The former is likely to cause problems for BDMs that use more standard supervised learning techniques (*e.g.*, those that use Linear Models or Decision Trees for their hypothesis class). The later is likely to cause problems for BDMs that model the data distribution (*e.g.*, those that use Gaussian Kernels for their hypothesis class). As a result, it seems that being able to identify blindspots of varying sizes and densities is a potentially important and challenging goal for BDM design.

The boundaries of a blindspot can be “fuzzy” in the sense that there may be images that do not belong to the blindspot that are both close, in the image representation space, to images in the blindspot and that have higher error. For example, looking at Figure 19, we can see that the model’s confidence decreases relatively smoothly as we approach the region that reflects “baseball gloves with people, but not in a baseball field.” However, if we look at the region around this region (Figure 20), we see that these images are almost all in baseball fields. As a result, it seems that being able to handle this fuzziness is a potentially important and challenging goal for BDM design.

F.1 PRELIMINARIES

Before presenting the details of our results, there are two questions to address.

What blindspots do we study? We consider blindspots discovered by prior works (Hohman et al., 2019; Sohoni et al., 2020; Singh et al., 2020; Plumb et al., 2022; Singla et al., 2021) as well as novel blindspots (Adebayo et al., 2022). Note that these prior works:

- Are not all explicitly about blindspots (*e.g.*, some consider “spurious patterns” which imply the existence of blindspots).
- All consider a model that was trained with standard procedures (*e.g.*, no adversarial training) on a real dataset (*e.g.*, no semi-synthetic images or sub-sampling the dataset).
- All consider blindspots that pertain to a single class.

How are our visualizations produced? To start, we use scvis (Ding et al., 2018) to learn a 2D embedding of the model’s representation of all of the training images of the class of interest. Because scvis combines the objective functions of tSNE and an AutoEncoder, we expect similar images to be near each other and dissimilar images to be far from each other in this 2D embedding. Then, we visualize that embedding and color code the points by the model’s confidence in its predictions for the class of interest. As a result, blindspots should appear as groups of lighter colored points.

With this visualization, we start by first exploring what types of images were in different parts of the embedding. Then, given a specific type of image, we go back and highlight the part of the embedding that best represents that type of image. As a result, many of the figures we show are more like “summaries that support our conclusions” than “exhaustive descriptions of all the types of images we identified.”

F.2 SUMMIT (HOHMAN ET AL., 2019)

In Figure 1 of their paper, Hohman et al. (2019) show that their method reveals that the model is relying on “hands holding fish” to detect “tench.”

- Dataset: ImageNet
- Class: tench
- Blindspots: tench without people holding them

Results of inspecting the 2D embedding:

- The embedding also reveals this blindspot. However, our analysis suggests that this blindspot is too broadly defined because “people holding tench” (Figure 10) has similar performance to “tench in a net” (Figure 11) and that the performance drop comes from “tench underwater” (Figure 12). We confirmed hypothesis by finding images on Google Images that match these descriptions and measuring the model’s accuracy on them.
- Additionally, the embedding reveals mislabeled images (Figure 13).

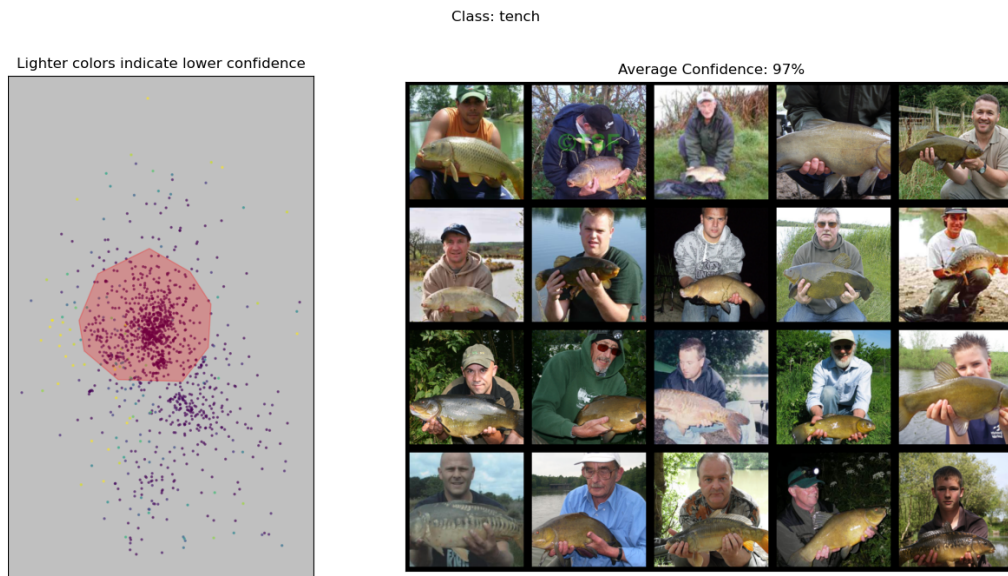


Figure 10: “tench with people holding them”

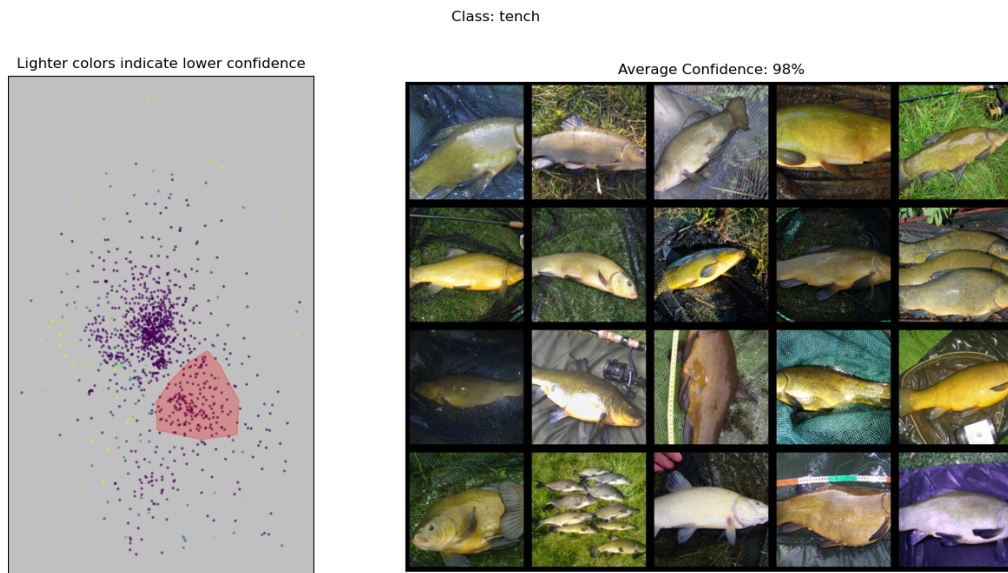


Figure 11: “tench in a net”

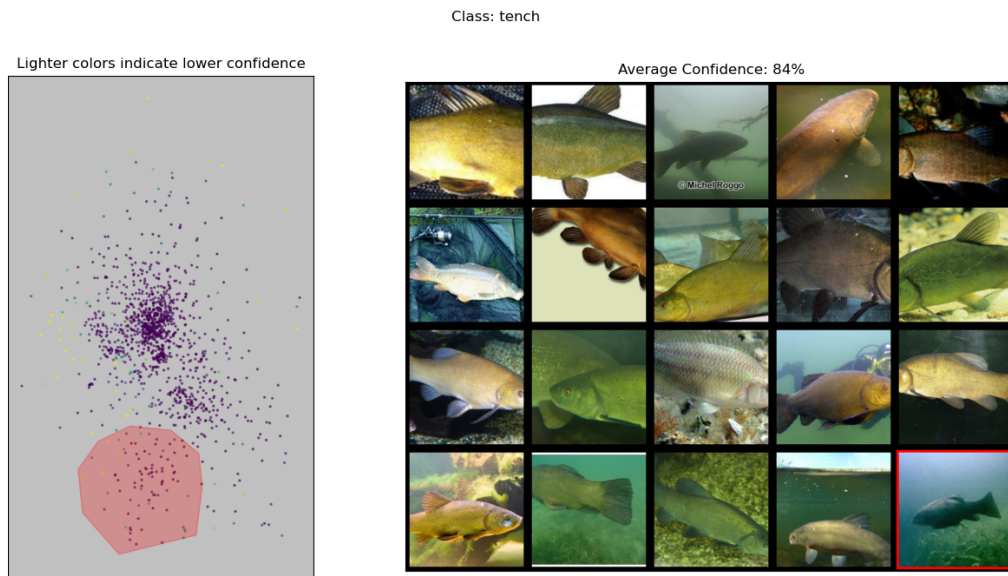


Figure 12: “tench underwater”

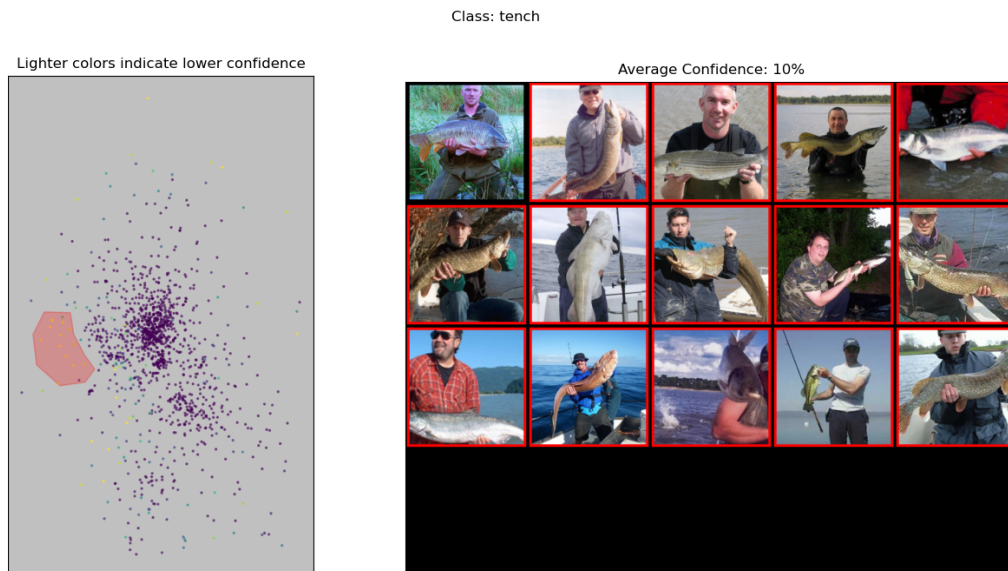


Figure 13: Mislabeled images

F.3 GEORGE (SOHONI ET AL., 2020)

In Appendix C.1.4 of their paper, Sohoni et al. (2020) discuss how their method is not effective at identifying one of the blindspots commonly studied in Group Distributionally Robust Optimization without using an external model’s representation for the image representation.

- Dataset: CelebA
- Class: blond hair
- Blindspot: men with blond hair

Results of inspecting the 2D embedding:

- By comparing Figure 14 to Figure 15, we can see that the embedding clearly reveals this blindspot.
- Additionally, Figure 16 shows that the embedding reveals an additional “blond women playing sports” blindspot.

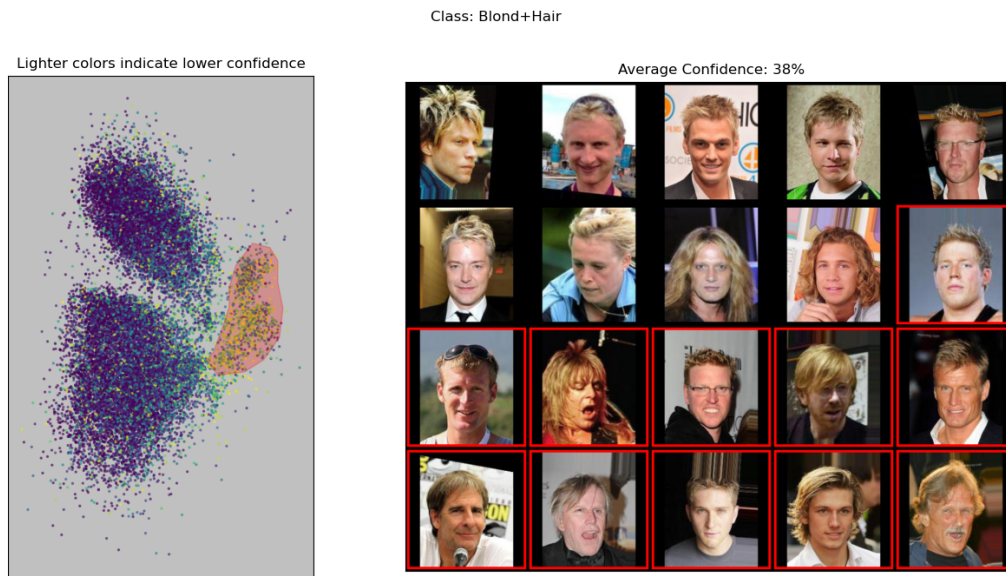


Figure 14: “blond men”

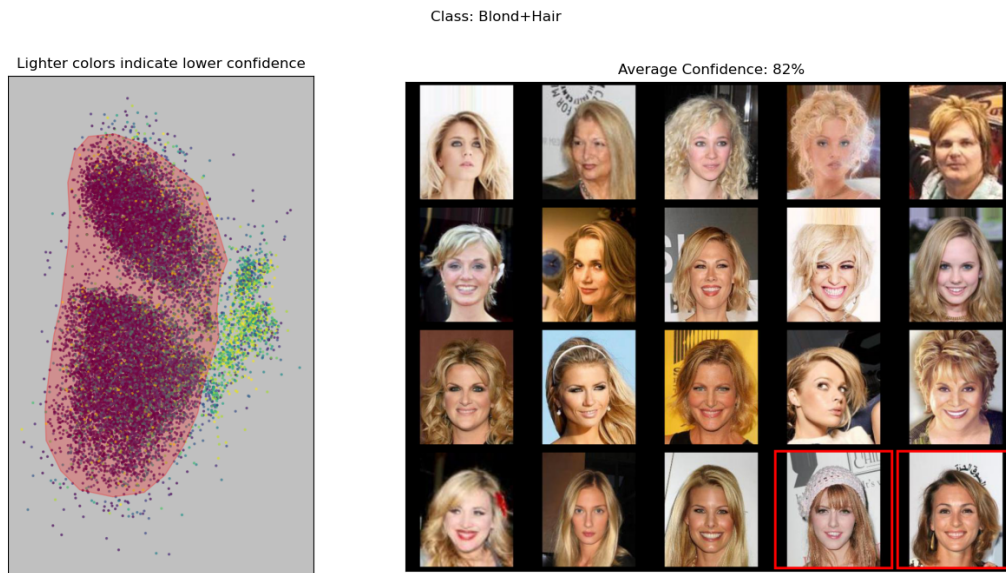


Figure 15: “blond women”

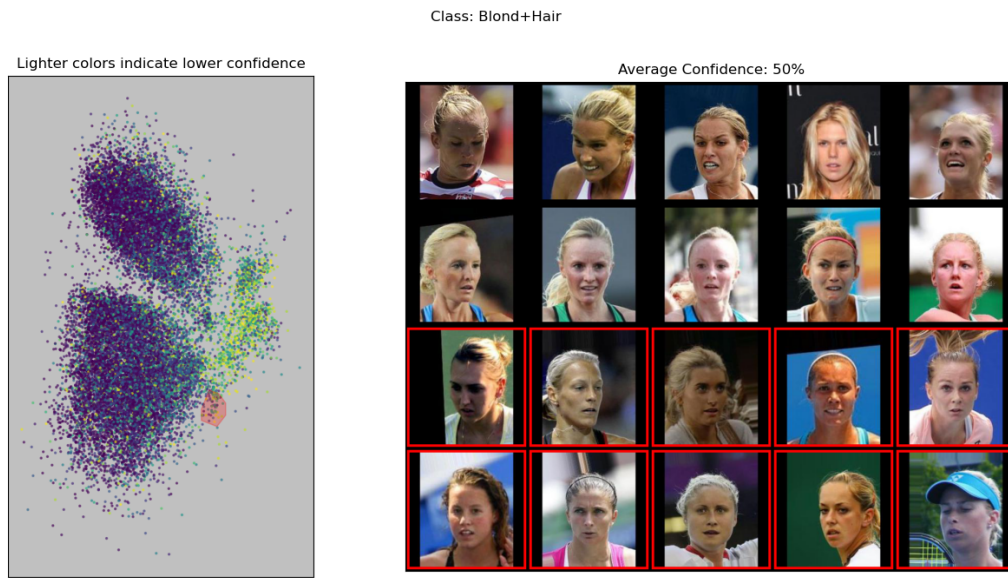


Figure 16: “blond women playing sports”

F.4 FEATURE SPLITTING (SINGH ET AL., 2020)

In Table 6 of their paper, Singh et al. (2020) organize the spurious patterns that they identify by their own measure of bias. We consider both the most and least biased spurious patterns from that table.

Most biased.

- Dataset: COCO
- Class: baseball glove
- Blindspot: baseball gloves without people

Results of inspecting the 2D embedding:

- Comparing Figure 17 to Figure 18, we see that the embedding also reveals this blindspot.
- However, we can also see that the model’s performance is not uniform across different types of images of “baseball gloves with people.” In particular, the embedding reveals two other blindspots (Figures 19 and 21)

Least biased.

- Dataset: COCO
- Class: cup
- Blindspot: cups without dining tables

Results of inspecting the 2D embedding:

- Comparing Figure 22 to Figure 23, we see that the embedding also reveals this blindspot.
- However, we can also see that the model’s performance is not uniform across different types of “cups without dining tables.” Figures 24, 25, and 26 show a few examples of the additional blindspots the embedding reveals.
- These results highlight an interesting subtlety. The cups in images with dining tables take up a greater portion of the image than the cups in images of sports stadiums, which raises the question: “Does this blindspot exist solely because the model is relying on ‘context’ that it shouldn’t be? Or is the standard image pre-processing making the cups too small (in terms of their size in pixels) to detect?”

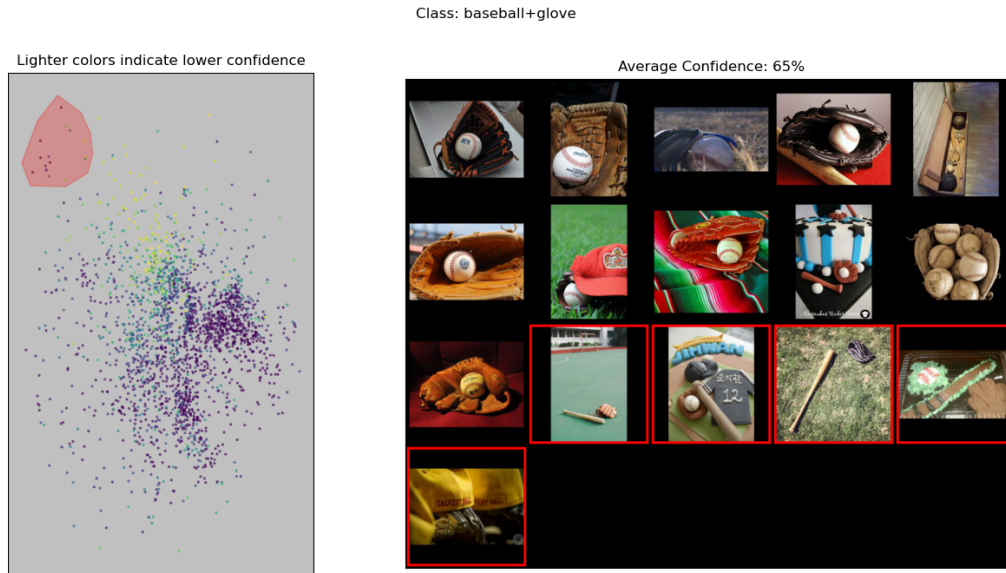


Figure 17: “baseball gloves without people”

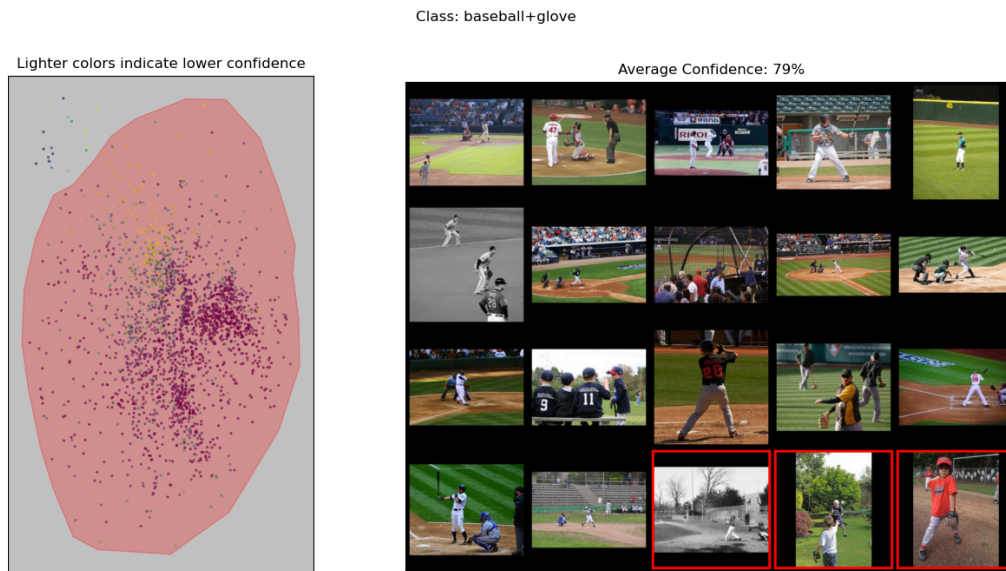


Figure 18: “baseball gloves with people”



Figure 19: “baseball gloves with people, but not in a baseball field”



Figure 20: If we look at the region around the “baseball gloves with people, but not in a baseball field” region (Figure 19), we see that the model’s confidence is still lower despite the fact that these images still are in baseball fields.

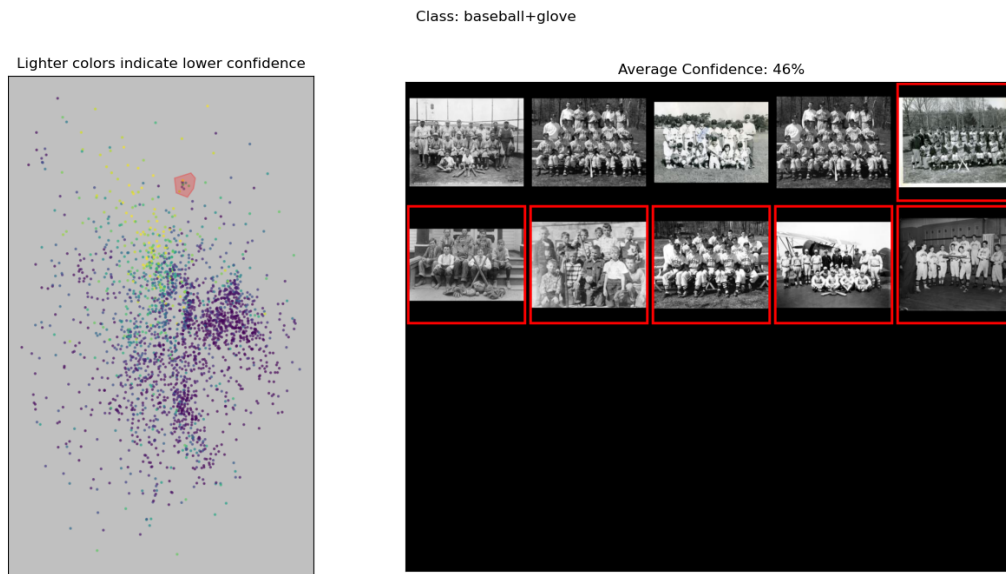


Figure 21: “black and white images of baseball teams”



Figure 22: “Cups without dining tables”

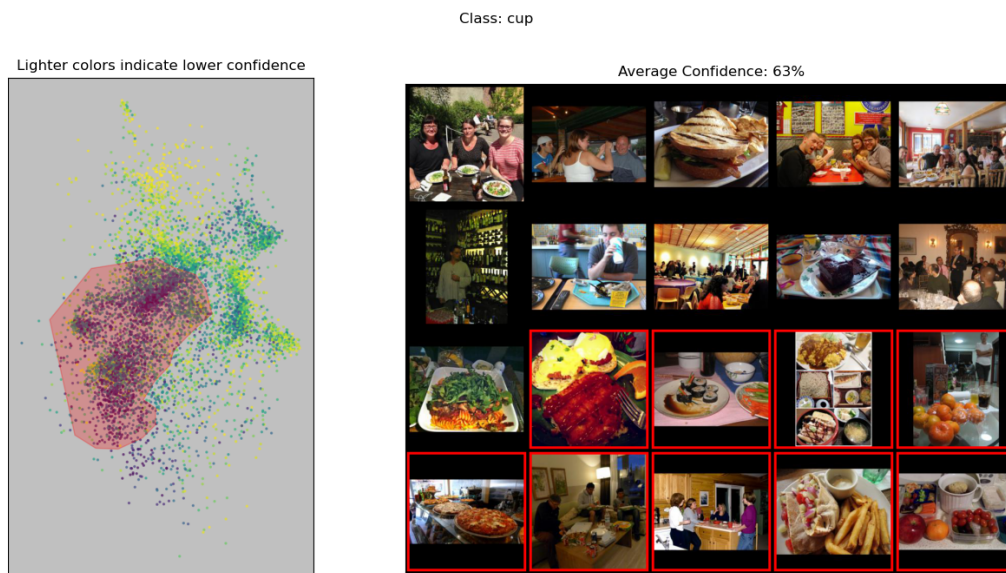


Figure 23: “Cups with dining tables”

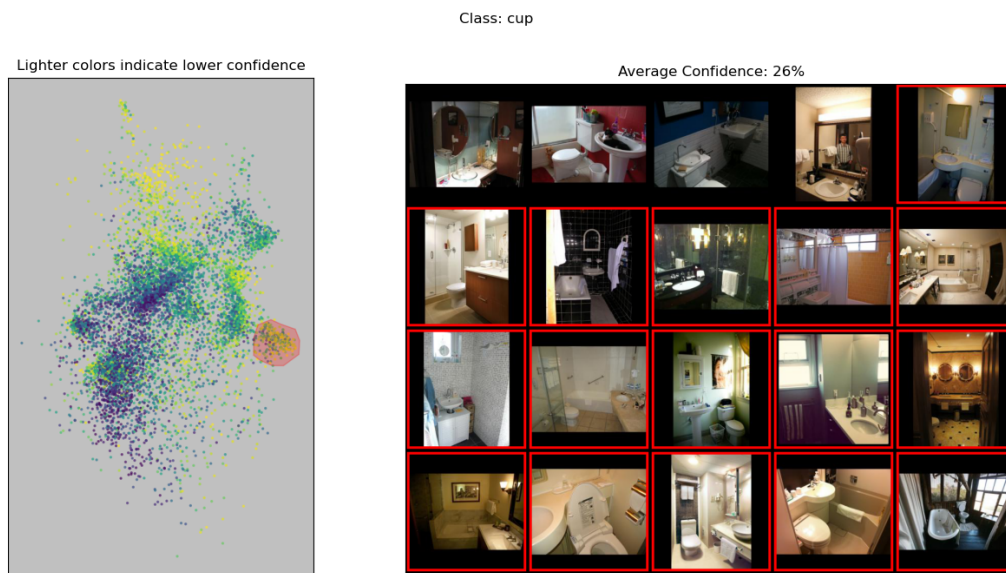


Figure 24: “Cups in bathrooms”

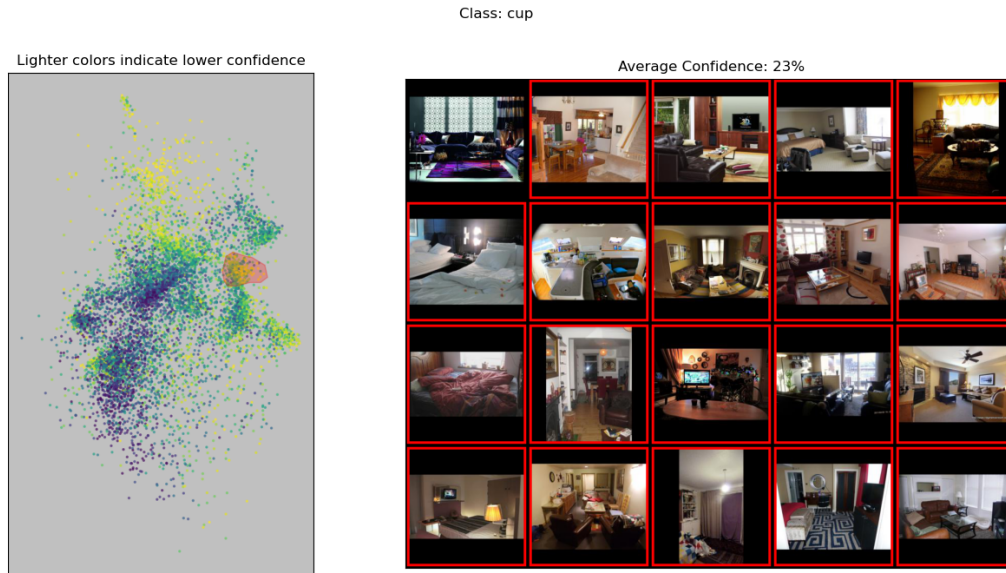


Figure 25: “Cups in living rooms or bedrooms”

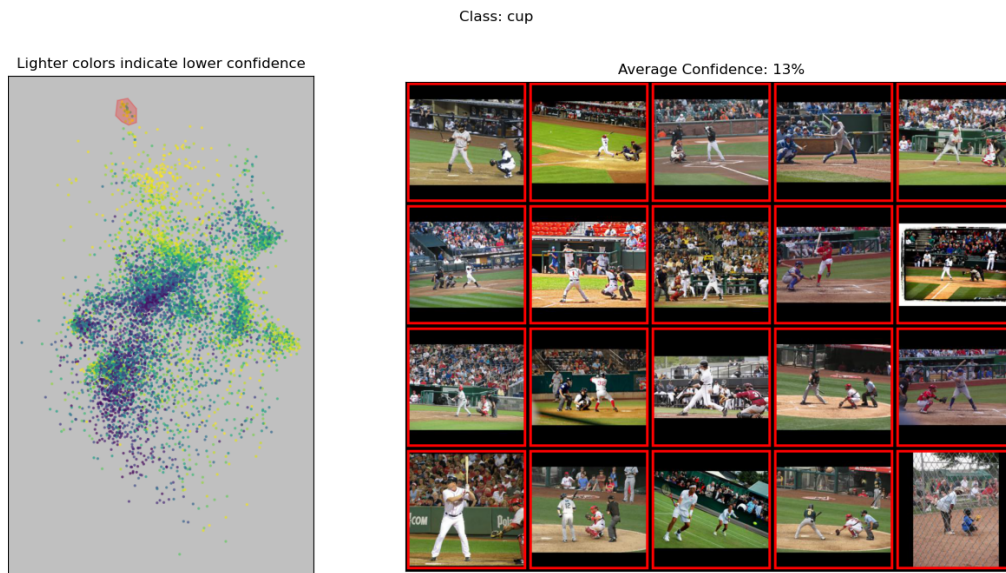


Figure 26: “Cups in sports stadiums”. Is the model using ‘context’ that it shouldn’t be? Or is the image pre-processing making detecting these cups very hard by compressing them to only a few pixels? Or is this a limitation of the model architecture?

F.5 SPIRE (PLUMB ET AL., 2022)

Plumb et al. (2022) claim that, compared to past works on identifying spurious patterns, that theirs is the first to identify negative spurious patterns. So we explore the example of a negative spurious pattern that they give. Additionally, they note that this class is part of two spurious patterns.

- Dataset: COCO
- Class: tie
- Blindspot: ties without people
- Blindspot: ties with cats

Results of inspecting the 2D embedding:

- Comparing Figure 27 to Figure 28, we see that the embedding reveals the “ties without people” blindspot.
- Comparing Figure 29 to Figure 30, we see that the embedding reveals the “ties with cats” blindspot. Note that this is a subset of the “ties without people” blindspot.
- Figures 31 and 32 show a few examples of the additional blindspots the embedding reveals.



Figure 27: “ties without people”

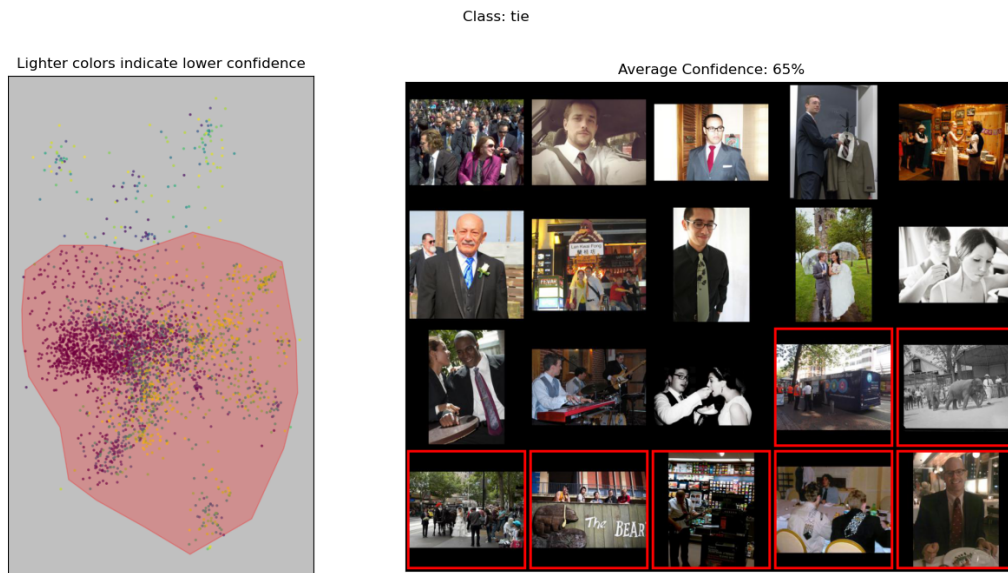


Figure 28: “ties with people”



Figure 29: “ties with cats”

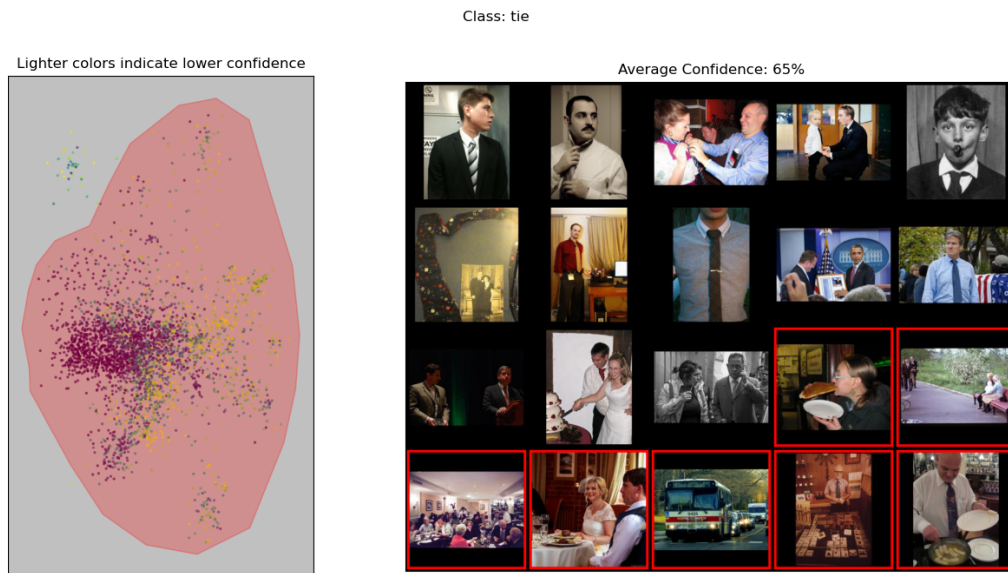


Figure 30: “ties without cats”

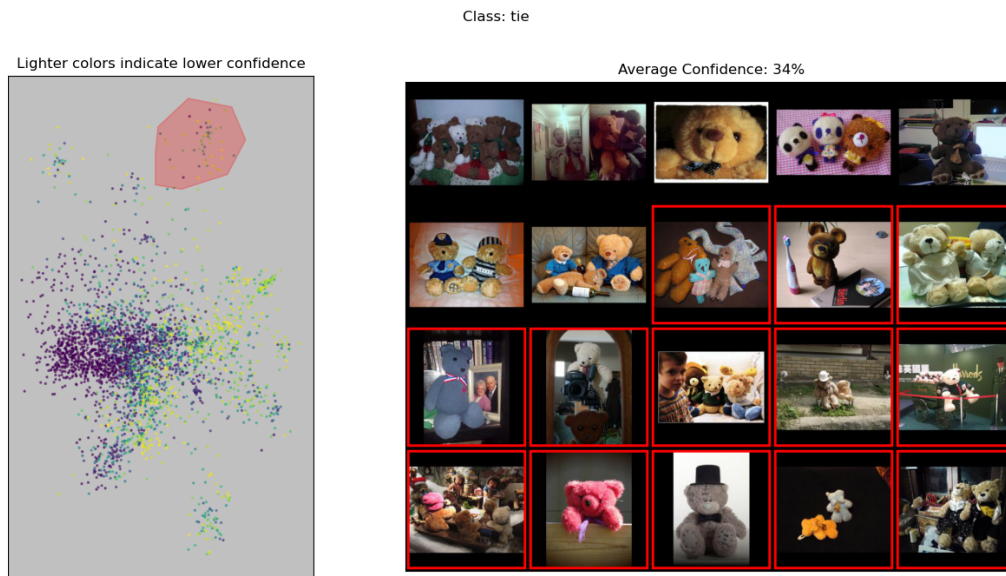


Figure 31: “ties with teddybears”

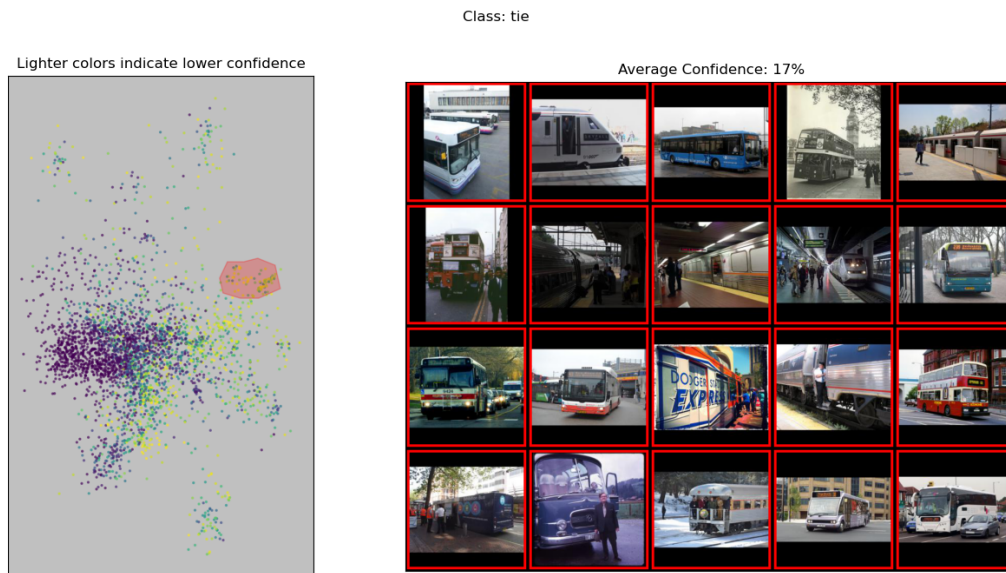


Figure 32: “ties with buses” Is the model using ‘context’ that it shouldn’t be or is the image pre-processing making detecting these ties very hard (the bus driver is frequently wearing a tie, but that tie is very small)?

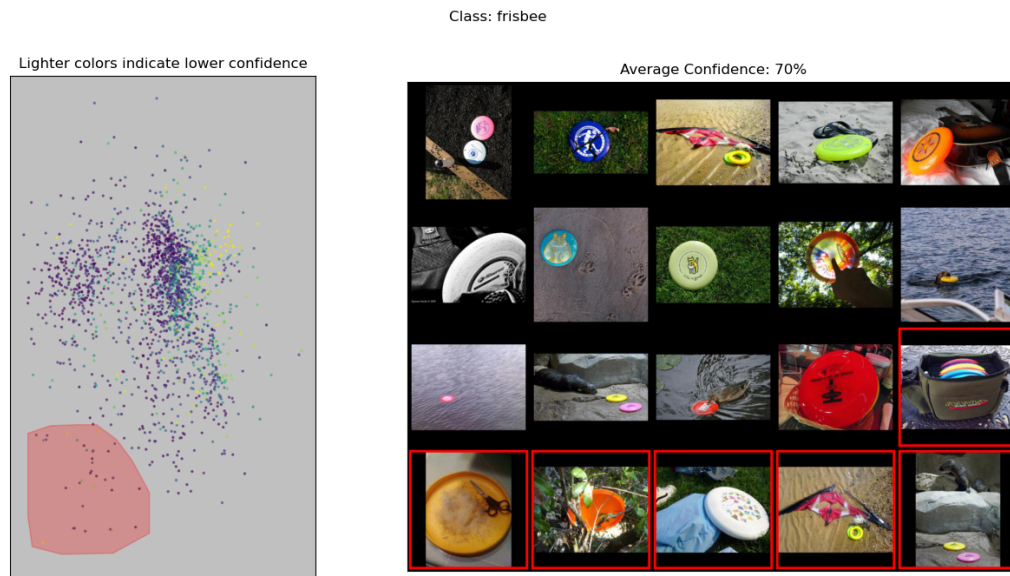


Figure 33: Additionally, when we see that the embedding also reveals the “frisbees without dogs or people” blindspot from (Plumb et al., 2022). However, these images look more like outliers than a proper cluster.

F.6 BARLOW (SINGLA ET AL., 2021)

In Table 2 of their paper, Singla et al. (2021) summarize some of the blindspots that they identify. We consider two of them. The first is “hog”, which is the class with the largest ALER-BER score (which they use as a measurement of how important a blindspot is). The second is “tiger cat”, which is the class with the largest increase in error rate for images in the blindspot.

Largest ALER-BER Score.

- Dataset: ImageNet
- Class: hog
- Blindspot: not “pinkish animals”

Results of inspecting the 2D embedding:

- By comparing Figure 34 to Figure 35, we can see that the embedding reveals this blindspot.

Largest increase in error rate.

- Dataset: ImageNet
- Class: tiger cat
- Blindspot: not “face close up”

Results of inspecting the 2D embedding:

- It does not appear that the embedding reveals this blindspot.
- By comparing Figure 36 to Figures 37 and 38, we can see that the embedding reveals a “grey tiger cats inside” blindspot.
- Looking at Figure 39, we can see that the embedding reveals mislabeled images.

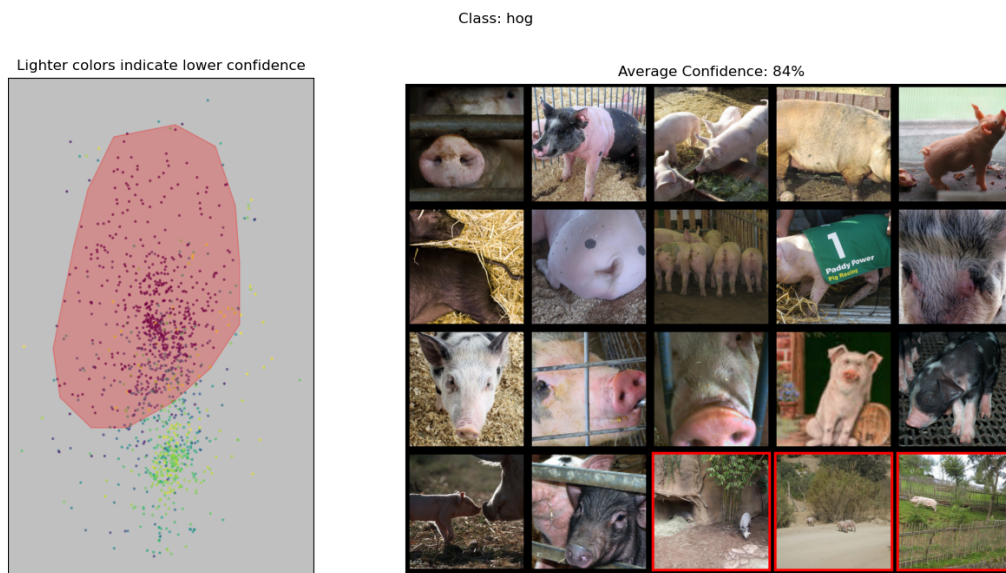


Figure 34: “pink hogs”

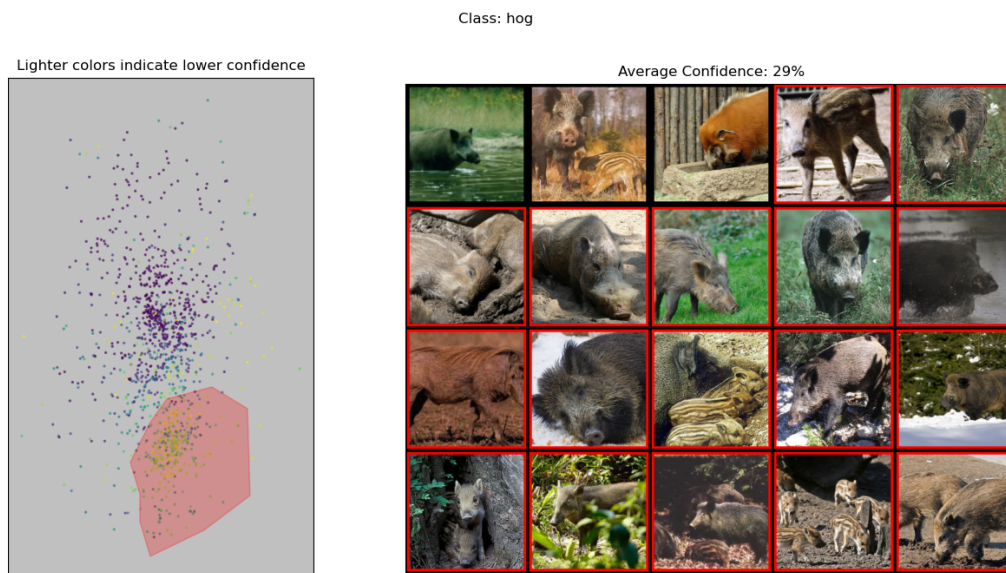


Figure 35: “brown hogs”

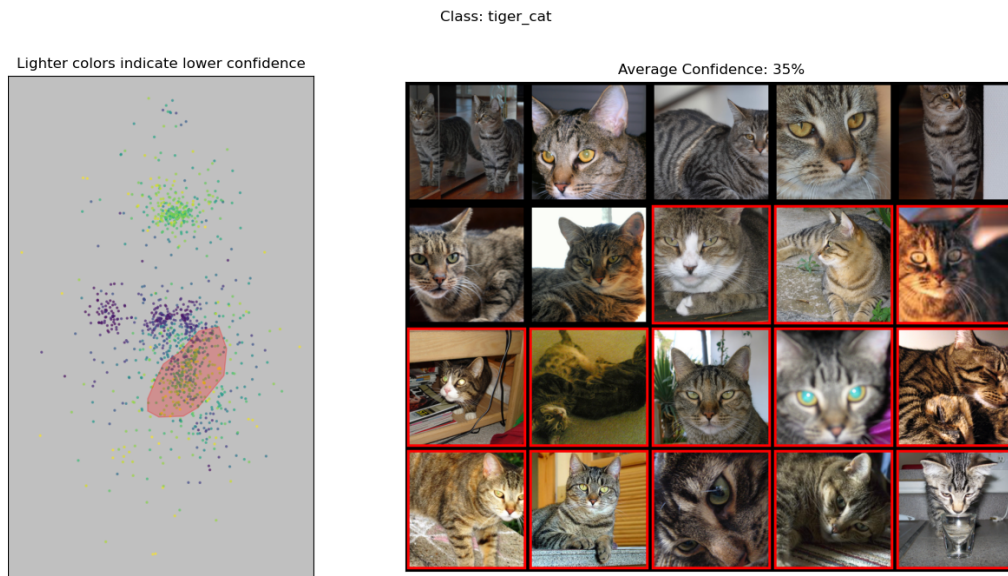


Figure 36: “grey tiger cats inside”

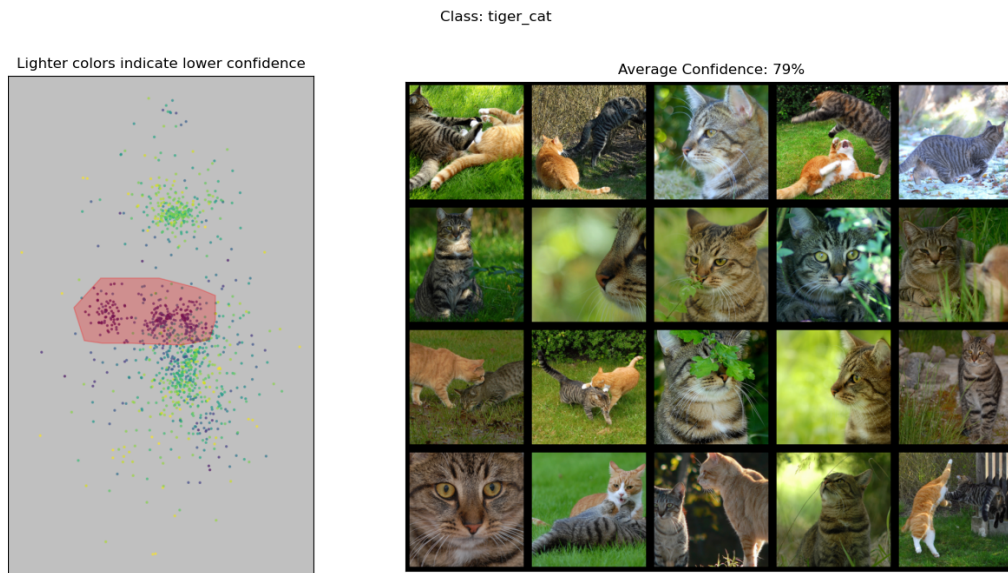


Figure 37: “grey tiger cats outside”

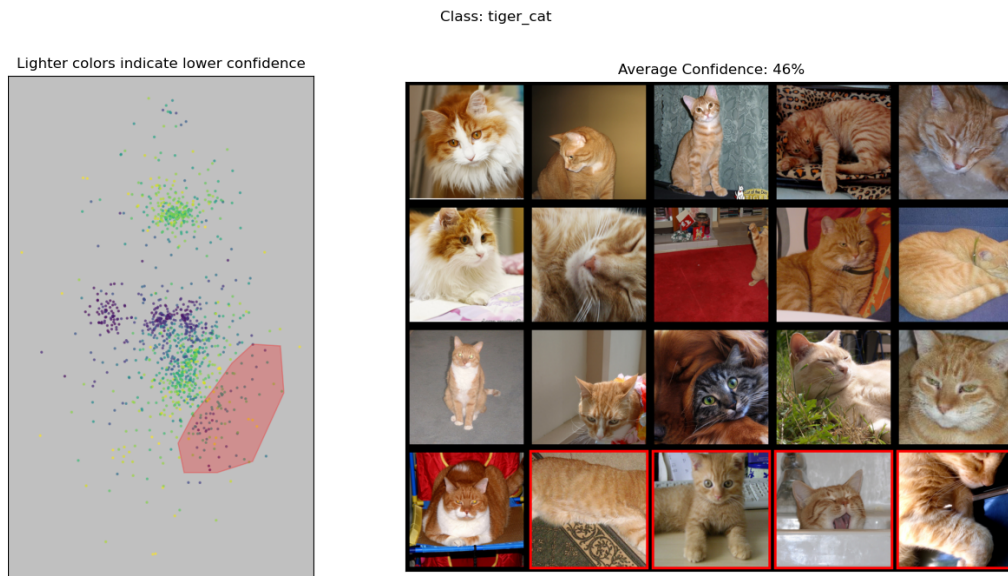


Figure 38: “orange tiger cats”

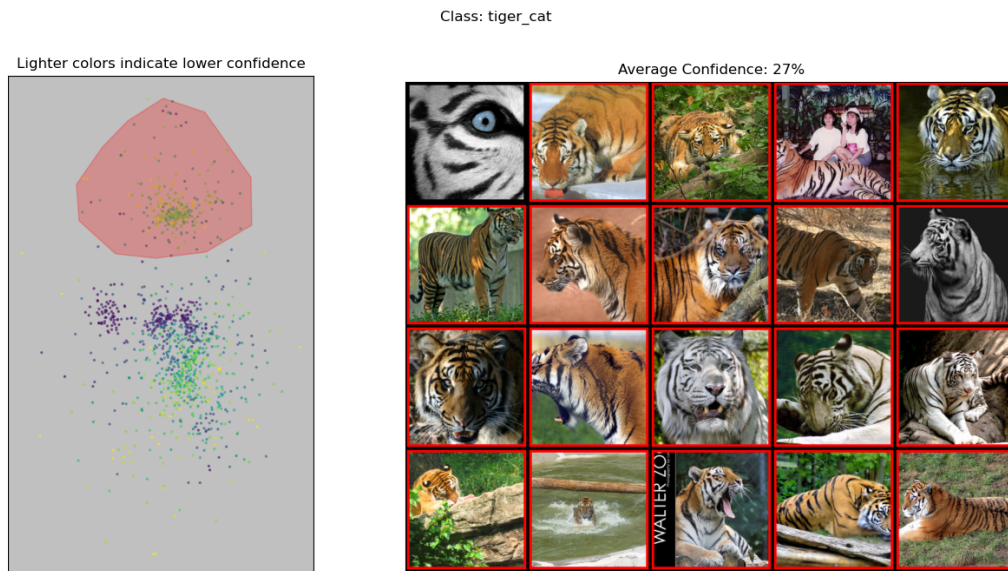


Figure 39: Mislabeled Images

F.7 NEW BLINDSPOTS

Because it has been observed that it is easier to find “bugs” in models when we know what to look for (Adebayo et al., 2022), we also explore classes that have not been discussed in prior work. We asked a colleague to pick a class from the list of ImageNet classes and they chose “library” because they thought that it seemed related to or correlated with other ImageNet classes.

- Dataset: ImageNet
- Class: library
- Blindspot: images of the outside of libraries
- Blindspot: images of a single bookshelf

Results of inspecting the 2D embedding:

- Figure 40 shows that the model is less able to label an image as a library when the image is of the outside of a library. Interestingly, this is even true when the building is labeled as a library.
- By comparing Figure 41 to Figures 42 and 43, we see that, without the context clues of “people” or “multiple bookshelves” the model is less able to label an image as a library when the image contains only a single bookshelf. This may be a data labeling problem because “bookcase” is another class in ImageNet.

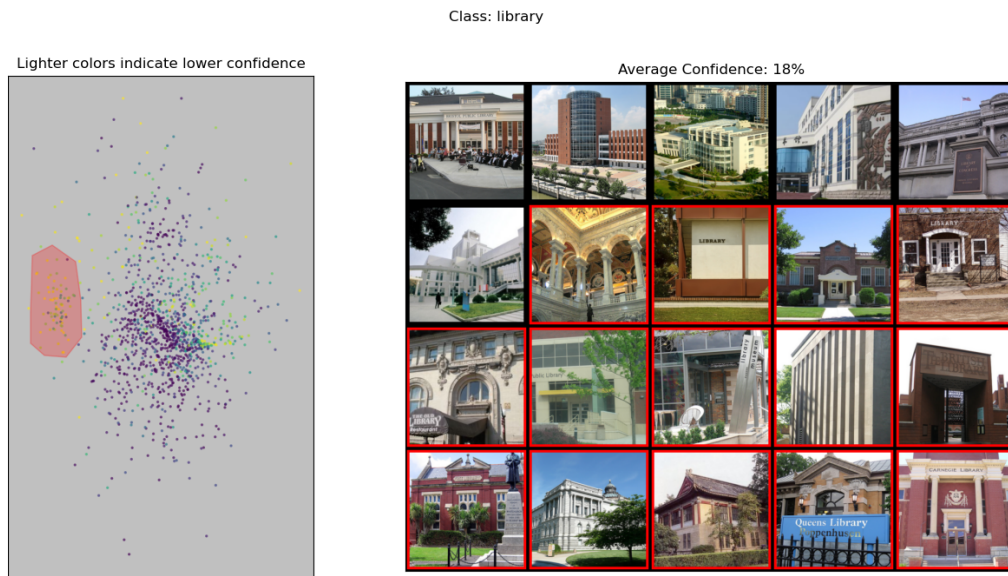


Figure 40: “outside of libraries”



Figure 41: “single bookshelf”



Figure 42: “people near bookshelves”

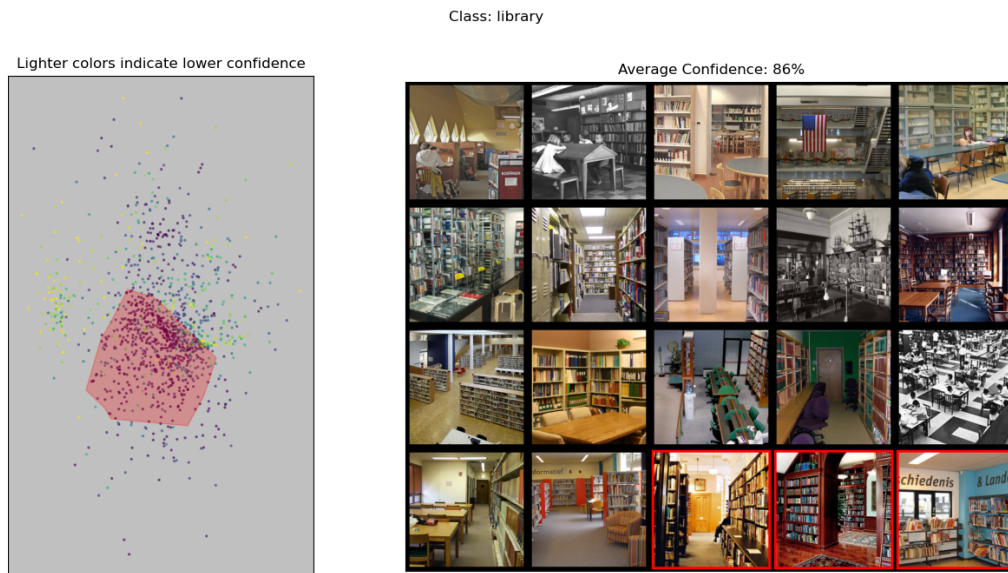


Figure 43: “multiple bookshelves without people”