## Appendix Overview

- In Appendix A, we present additional details on the AdaVision system as described in Section 3, including details about adaptive test retrieval and automatic test labeling (A.1), as well as adaptive topic generation based on templates and user topics (A.2).
- In Appendix B, we include additional details about the user studies in Section 4.2, as well as details about the statistical analyses.
- In Appendix C, we expand upon the comparison of AdaVision with Domino (Section 4.3 in the main text), providing an explanation of Domino and its hyperparameters, our criterion for coherency, a list of all slice descriptions (topics) from Domino used during evaluation, and a list of the 30 user-found AdaVision topics we compared Domino against.
- Finally, in Appendix D, we discuss hyperparameters used for finetuning in Section 4.4, list all control and treatment topics, and include an additional evaluation measuring whether finetuning on treatment topics improves other, conceptually unrelated bugs.

## A. Additional details on AdaVision

In Section 3, we described AdaVision, which includes a test generation loop that retrieves images with CLIP and an topic generation loop that generates topics with GPT-3. Here, we expand on the details of both loops.

### A.1. Test generation loop

In the test generation loop, users explore a candidate topic $t$. Each iteration of the loop, AdaVision retrieves test suggestions relevant to the topic, and, when possible, automatically imputes pass/fail labels for these suggestions in order to minimize user labeling effort. In this section, we provide additional details about the retrieval and labeling steps of a single iteration of the test generation loop.

Recall that $m$ is our target vision model (*e.g.* a classification model), and $m(x)$ is the model output on an image $x$ (*e.g.* the label string, such as "banana"). At any given iteration of the test generation loop, we have a textual topic description $z$ and a (possibly empty) set of already labeled tests $\mathcal{D}$, where each test is a triplet $(x, m(x), y)$. The label $y \in \{-1, 1\}$ refers to whether the test failed (-1) or passed (1). We may also have a set of previously reviewed, off-topic tests $\mathcal{D}_{\text{off-topic}}$. In Algorithm 1, we step through the retrieval and labeling steps of the iteration. For space, we expand on two of the steps in the algorithm box below.

**Sampling previous tests $x_1, x_2, x_3$ for retrieval.** In the retrieval step, we incorporate three previous tests from $\mathcal{D}$. Each of the three tests is sampled from a categorical distribution over $\mathcal{D}$, where each $x_j \in \mathcal{D}$ has probability $p_j$ of being selected. We prefer to select tests where the model *confidently* fails over tests where the model less confidently fails, which are still preferred over passed tests. Thus, when available, we use the model's *prediction confidence* $s_m(x_j) \in [0, 1]$ for each example $x_j \in \mathcal{D}$ to compute $p_j$. For example, in classification, $s_m$ is the model's maximum softmax score. If this value is unavailable (*e.g.* for some commercial APIs), we fix $s_m(x_j) = 1$ for all examples. To compute $p_j$, we compute a score $\alpha_j = 1 - y_j s_m(x_j)$ per example (note that this is 0 when the test is confidently correct and 2 when the model is confidently incorrect), and we set $p_j$ to be the normalized version of $\alpha_j$.

**Lightweight classifiers $f, f_{\text{off-topic}}$ for automatically labeling tests.** When $|\mathcal{D}| > 0$, we use two lightweight classifiers to automatically impute whether tests have passed, failed, or are off-topic. The lightweight classifiers are specialized to the current topic (*i.e.* we train new classifiers for new topics). Functionally, a lightweight classifier is a Support Vector Classifiers (SVC). One lightweight classifier $f$ maps $(x, m(x))$ to predicted pass/fail label in $\{-1, 1\}$. Specifically, the tuple $(x, m(x))$ is transformed into a single vector by first embedding both $x$ (image) and $m(x)$ (string) with CLIP ViT-L/14, followed by concatenating the two into $[\text{CLIP}(x), \text{CLIP}(m(x))]$. This vector is used as the feature representation of the SVC. New tests are re-sorted according to the prediction of $f$ and $f$'s confidence, with likely failures shown first. Classifier $f_{\text{off-topic}}$ operates on the same representation as $f$ for each test, but instead predicts binary labels for in-topic / off-topic. These classifiers take less than a second to train and run, so we re-train them at each iteration of the test generation loop.

### A.2. Topic generation loop

In the topic generation loop, users collaborate with AdaVision to generate candidate topics to explore using GPT-3. We generate candidate topics in two phases. In the first phase, we prompt GPT-3 using a pre-written set of templates and collect the completions. We share the set of prompt templates used in Listing 1, replacing {LABEL} with predefined label names or existing user topics (*e.g.* `stop sign`). In

**Algorithm 1:** Iteration of the test generation loop.

```
Input: Textual topic description $z$, previously labeled tests $\mathcal{D} = \{(x, m(x), y)\}$, previous off-topic tests
       $\mathcal{D}_{\text{off-topic}}$
Compute $q_t \leftarrow \text{CLIP}(z)$                                                          ▷ Figure 2A
if $|\mathcal{D}| > 0$ then
    Sample $x_1, x_2, x_3 \sim \text{Categorical}(|\mathcal{D}|, p_j)$, where $p_j$ is computed according to the text in A.1
    Aggregate $q_i \leftarrow \sum_k \beta_k \cdot \text{CLIP}(x_k)$, with $\beta \sim \text{Dirichlet}(1, 1, 1)$
    Set $q \leftarrow \text{slerp}(q_t, q_i, r)$, with $r \sim \text{Uni}(0, 1)$
else
    Set $q \leftarrow q_t$
end
Retrieve approximate nearest neighbors of $q$ from LAION-5B                                     ▷ Figure 2B
Exclude retrievals whose CLIP image embeddings have cosine similarity $> 0.9$ with any previous test $x \in \mathcal{D}$
Collect model outputs for all retrieved images to obtain new collection of tests $\mathcal{S} \leftarrow [(\tilde{x}, m(\tilde{x}))]$
if $|\mathcal{D}| > 0$ then
                                                                                               ▷ Figure 2C
    Train a lightweight classifier $f$ on previously labeled tests $\mathcal{D}$ as described in A.1
    Sort $\mathcal{S}$ according to $f(\tilde{x})$ for $\tilde{x} \in \mathcal{S}$, placing predicted fails far from the decision boundary first, and
      predicted passes far from the decision boundary last Update $\mathcal{S}$ to contain $(\tilde{x}, m(\tilde{x}), f(x))$, so that we
      can display the imputed label to the user
    Train a second lightweight classifier $f_{\text{off-topic}}$ to differentiate between previous in-topic tests $\mathcal{D}$ and
      previous off-topic tests $\mathcal{D}_{\text{off-topic}}$
    Place tests $\tilde{x} \in \mathcal{S}$ for which $f_{\text{off-topic}}(x)$ predicts "off-topic" at the end of $\mathcal{S}$
end
return sorted $\mathcal{S}$ to the user for confirmation / correction.                               ▷ Figure 2D
```

the second phase, we gather suggestions from the first round, append previously explored topics with high failure rates, and gather a second round of suggestions. We place topics with the highest failure rates at the end of the prompts to account for GPT-3's recency bias [5]. Finally, topics are presented to users, who explore ones they deem interesting and important.

### A.3. Web interface

We provide screenshots of the AdaVision web interface in Figure 7. The topic generation loop is represented as a root page that suggests topics to explore, and individual topics are represented as folders (Figure 7 left). Tests within folders are represented as rows mapping images to model outputs (Figure 7 right).

## B. Additional details for user studies

In Section 4.2, we described a large set of user studies used to evaluate AdaVision's ability to enable users to find bugs in state-of-the-art vision models. Here, we include additional details about the study setups and statistical analyses.

Listing 1: Example prompts used in topic generation loop.

```
List some unexpected places to see a {LABEL}
List some places to find a {LABEL}
List some other things that you usually find
    with a {LABEL}
List some artistic representations of a {LABEL}
List some things that can be made to look like
    a {LABEL}
List some types of {LABEL} you wouldn't normally
    see
List some dramatic conditions to photograph
    a {LABEL}
List some conditions a {LABEL} could be in that
    would make it hard to see
List some things that are the same shape as a
    {LABEL}
List some {LABEL} that are a different color than
    you would expect
```

**Study setup.** All participants undertook the study virtually in a single 60-minute Zoom session. At the start of the session, participants were shown a 5-minute video introducing how to use the AdaVision web interface. Next, the experimenter walked through instructions for the testing task: as described in the

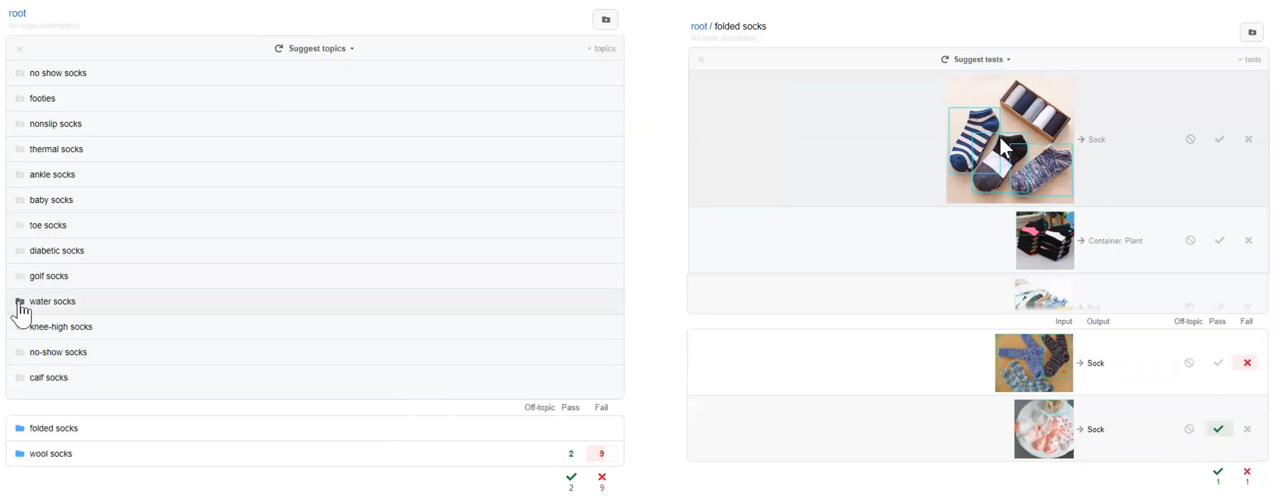


Figure 7: In the AdaVision web interface, topics are represented as folders, and tests are represented as rows mapping images to model outputs. The topic generation loop is represented as a root page that suggests topics to explore (left), while the test generation loop within each topic suggests tests that lightweight classifiers label as potential failures (right, top panel).

main text, participants were instructed to find as many failure-prone topics (bugs) for a specific category (*e.g.* banana) as possible, and to switch topics whenever they found more than a threshold of failures within a topic. This latter instruction prevented users from endlessly exploiting a topic to inflate the total failure count. We adjusted the threshold at which a topic became a bug (and users should move on) based on the time users had for testing: for classification and image captioning, users tested models for 20 minutes with a bug threshold of 10 failures, while for object detection, users tested the model for 15 minutes with a bug threshold of 8.

When introducing the task, the experimenter defined failed tests as follows:

- For participants testing classification models, a test failed if the model predicted an object not present in the image. Users were instructed to look for failures among pictures of a specific category (*banana* or *broom*), *e.g.* failures among pictures of bananas (Figure 8). Participants were given class definitions from an ImageNet labeling guide used in [4].

- For those testing object detection models, a test failed if the model failed to box any instance of the given category (*bicycle* or *stop sign*), *e.g.* as shown in Figure 9.

- Participants testing image captioning models were asked to imagine that they were testing a product used by visually impaired customers to caption everyday scenes. The participants' task was to find images (tests) for which the model produced false or incorrect captions, excluding counting, color, and gender or age mistakes (Figure 10). Participants looked for such tests among pictures of a specific category (*i.e.* scenes customers might encounter in a *kitchen* or *elementary school*).

The experimenter then asked each user to practice using the web interface by testing the model a third, held-out object or location for 10 minutes. For classification, this was a wine bottle; for object detection, this was a fire hydrant; and for image captioning, this was a *garden*. After two rounds of testing in the main experiment, the study concluded with an exit survey as described in Table 3. The study compensation was a $25 Amazon gift card.

**Additional results.** As discussed in Section 4.2, AdaVision helped users find significantly more failing tests than NonAdaptive, with significance determined by paired t-tests in each task. In classification, $t(16) = 2.27, p < 0.05$; in object detection, $t(16) = 3.42, p < 0.005$, and in image captioning, $t(8) = 2.56, p < 0.05$. These corresponded to normalized effect sizes of $d = 0.588$ in classification, $d = 0.882$ in object detection, and $d = 0.967$ in image captioning. We also counted the number of users who could find bugs during testing, *i.e.* identify a topic that hit the threshold number of fails (8 for object detection, 10 for the other tasks). Overall, 28/40 users found

## Classification Instructions

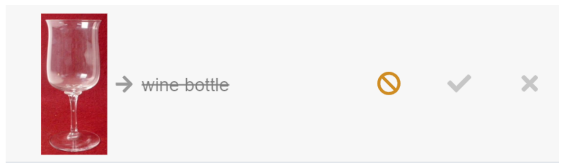
To help narrow down your search, we're only going to consider certain kinds of images: images which **contain an object Y.**

- ✅ A test **passes** if the predicted object *is actually in the image*.
- ❌ The test **fails** if the predicted object *is not in the image*.

### Examples

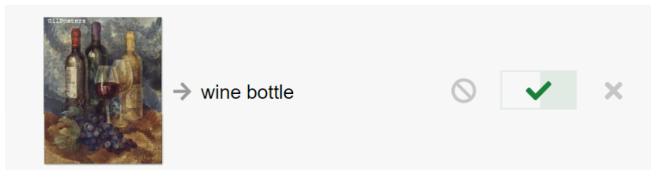
Let **object Y= "wine bottle"**.

First, we'll check whether we should mark the test as **off-topic.**

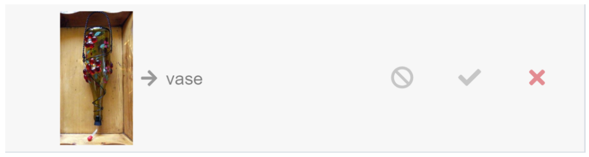


The image does not contain a wine bottle, so it is off-topic.

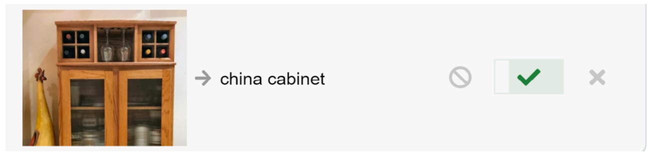
After checking if the test is off-topic, let's review when to mark a test as **pass**/**fail.**



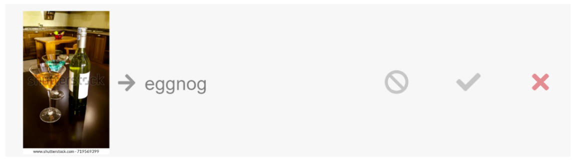
The image contains a wine bottle. The test **passes** because a wine bottle is in the image.



The image contains a wine bottle (upside down). The test **fails** because a vase is not in the image.



The image contains wine bottles (to the left and right of the wine glasses). The test **passes** because a china cabinet is in the image.



The image contains a wine bottle. The test **fails** because eggnog is not in the image.

## Should I be exploring many topics, or staying within one topic?

Your task is to **maximize the number of topics** (folders) that *each* have **10+ failures.**

- The round ends when the time is up.
- A good strategy is to explore many topics. (Don't stay in a topic longer than 10 fails!)

Figure 8: Example instructions for classification users.

## Detection Instructions

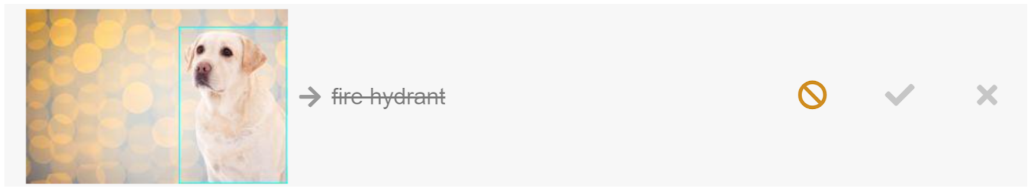
For the sake of the study, we're only interested in a certain kind of image:

- The image must contain one or more instances of **object Y**
- An image is a **fail** if the model does not locate **ANY** occurrences of **object Y**

## Examples

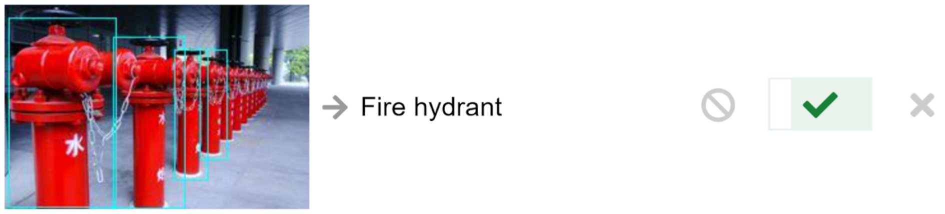
Let **object Y= "fire hydrant"**.

First, we'll check whether the image is relevant to the study.

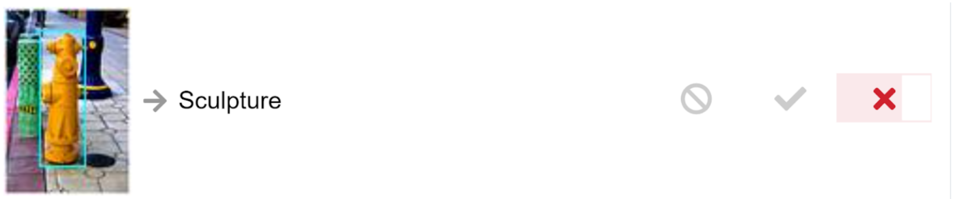


The image does not contain a fire hydrant, so it is off-topic.

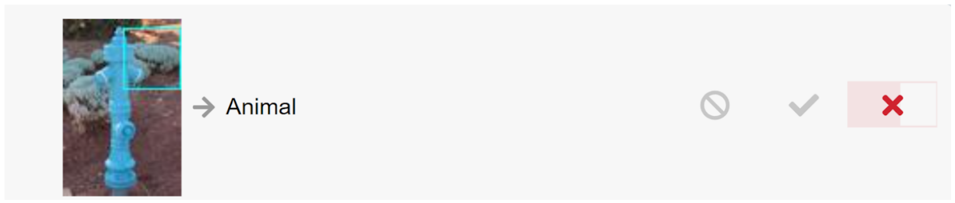
Next, we'll decide whether to mark a test as **pass/fail.**



The image contains a fire hydrant. The test **passes** because the model boxes and labels at least one of the fire hydrants. It doesn't matter that it doesn't box all of the hydrants.



The image contains a fire hydrant. Although the model boxes, the hydrant, it doesn't correctly label it, so the test **fails.**



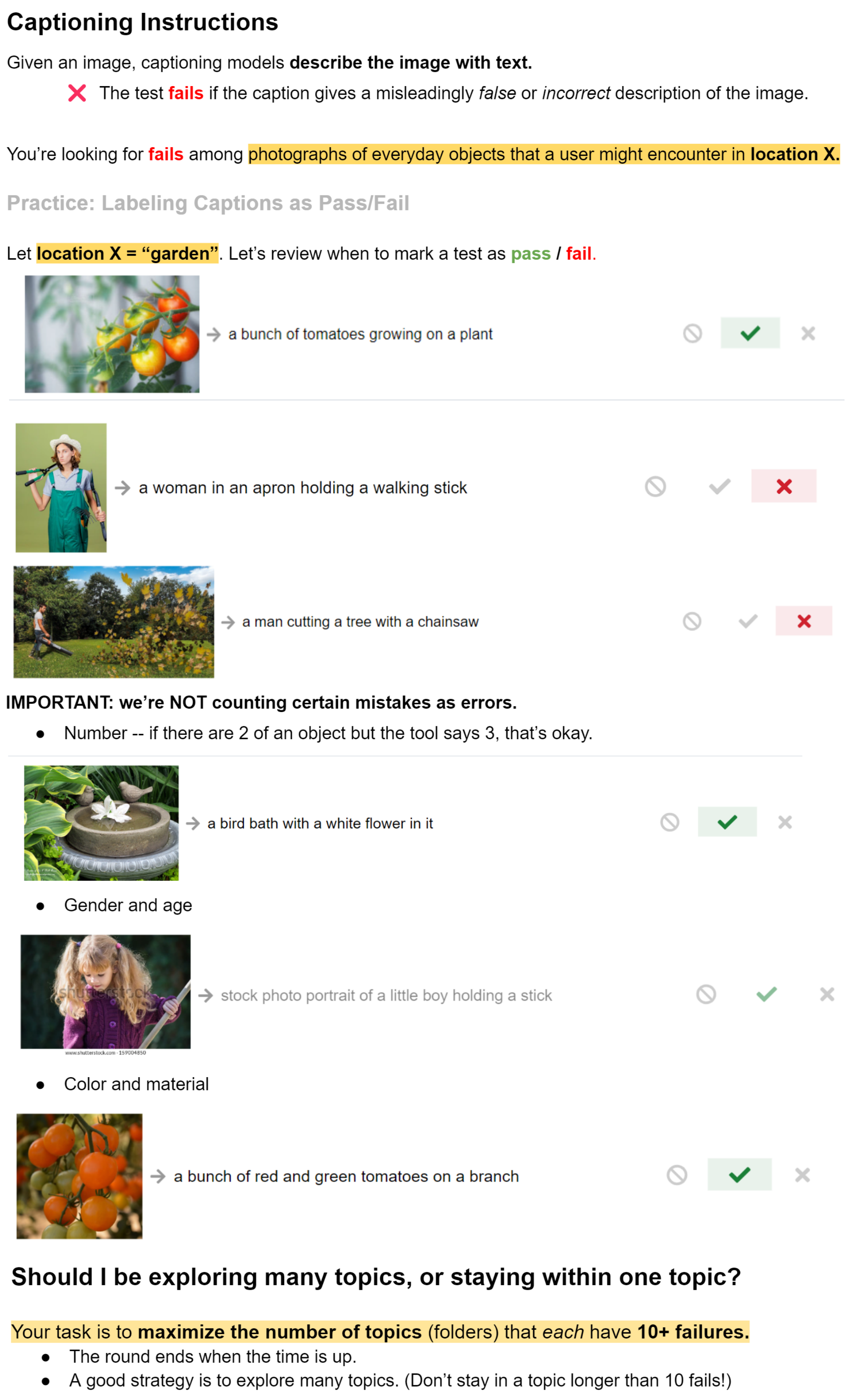
The image contains a fire hydrant. The test **fails** because the fire hydrant is not boxed and labeled.

**Should I be exploring many topics, or staying within one topic?**

Your task is to **maximize the number of topics** (folders) with **8+ failures.**

- The round ends when the time is up.
- A good strategy is to explore many topics. (Don't stay in a topic longer than 8 fails!)

Figure 9: Example instructions for object detection users.



## Captioning Instructions

Given an image, captioning models **describe the image with text.**

✗ The test **fails** if the caption gives a misleadingly *false* or *incorrect* description of the image.

You're looking for **fails** among photographs of everyday objects that a user might encounter in **location X.**

### Practice: Labeling Captions as Pass/Fail

Let **location X = "garden"**. Let's review when to mark a test as **pass** / **fail**.

→ a bunch of tomatoes growing on a plant ⊘ [✓] ✗

→ a woman in an apron holding a walking stick ⊘ ✓ [✗]

→ a man cutting a tree with a chainsaw ⊘ ✓ [✗]

**IMPORTANT: we're NOT counting certain mistakes as errors.**

- Number -- if there are 2 of an object but the tool says 3, that's okay.

→ a bird bath with a white flower in it ⊘ [✓] ✗

- Gender and age

→ stock photo portrait of a little boy holding a stick ⊘ ✓ ✗

- Color and material

→ a bunch of red and green tomatoes on a branch ⊘ [✓] ✗

### Should I be exploring many topics, or staying within one topic?

Your task is to **maximize the number of topics** (folders) that *each* have **10+ failures.**

- The round ends when the time is up.
- A good strategy is to explore many topics. (Don't stay in a topic longer than 10 fails!)

Figure 10: Example instructions for image captioning users.

| Question | Type |
|---|---|
| How difficult was it to find bugs in the first round? | 5-point Likert scale |
| How difficult was it to find bugs in the second round? | 5-point Likert scale |
| How useful was the web tool for finding bugs? | Multiple-choice {I could have found these bugs using existing error analysis tools I have access to, I could not have found these bugs using existing error analysis tools I have access to.} |
| Did you use topic suggestions? Were they helpful? | Multiple-choice {Yes, and they were helpful in generating topics that caused failures, Yes, and they were helpful in generating ideas for topics to explore, Yes, but they did not generate good ideas for topics to explore, No, I did not use topic suggestions.} |

Table 3: Items in exit survey.

bugs during testing with AdaVision, while only 16/40 could find such a high-failure topics in the baseline round. When surveyed about perceived difficulty of finding bugs, users also felt finding bugs was easier with AdaVision than without, with $t(40) = 4.18, p < 0.0005$. When surveyed about the helpfulness of GPT-3's topic suggestions, 24/34 users who used the topic suggestions marked that they were helpful for exploration.

## C. Additional details for comparison with automatic slice discovery

In Section 4.3, we compared AdaVision with Domino [1], showing that bugs found with AdaVision are more difficult. Here, we provide an explanation of the Domino method and its hyperparameters, define our criterion for coherency, list *all* slice descriptions (topics) from Domino used during evaluation (along with whether they satisfied coherency), and list the 30 user-found AdaVision topics.

**Details on** Domino**.** We use the official release of Domino [1], available at https://github.com/HazyResearch/domino. To generate slice proposals for a label in {banana, broom, candle, lemon, sandal, wine bottle}, we ran the target model (ViT-H/14 or ResNet-50) over ImageNet validation examples of that class. Then, we used Domino's error-aware mixture model to generate 5 slices (per class). The error-aware mixture model first generates $\bar{k}$ candidate slices (clusters of images) before selecting the best 5 slices. As in the original paper, we set $\bar{k} = 25$, and we initialize groups using the confusion matrix setting. Additionally, Domino uses a hyperparameter $\gamma = 10$ to control the weight placed on *incorrect* examples when slicing. In the original paper, the authors used $\gamma = 10$ for all datasets, which we also initially tried. However, we found that this setting produced slices that were too easy (no errors within the slices). Thus, we matched the authors' blog post applying Domino to ImageNet[1], conducting final experiments with $\gamma = 40$. (Larger values of $\gamma$ resulted in too much cluster incoherency.)

In our experiments, we evaluated two variations of Domino, which use the same image clusters but differ in their captioning strategy. The first, Domino (BERT), is the original proposal in [1]; to caption a cluster, Domino (BERT) samples 250,000 BERT or Wikipedia completions, per cluster, of a set of templated captions; we provide the templates we used in Listing 2. Each cluster is then matched to the caption with the highest cosine similarity in CLIP space. We note that in the original paper, the authors used only one template: "a photo of a [MASK]"; our modifications to this template account for the fact that all clusters are about a certain class (*e.g.* clusters of banana images). Thus, we enforce that the label (banana) appears in the caption by populating {LABEL} in the template with the appropriate class name. We also compared against our own variation of Domino, Domino (OFA), which uses Alibaba's OFA-huge to more coherently caption clusters. Here, we run OFA-Huge over all examples in a cluster and select the individual caption that maximizes cosine similarity with the cluster mean.

**Coherency.** When reporting failure rates for Domino topics, we calculated failure rates only over descriptions that were coherent. Descriptions were deemed incoherent if they were nonsensical (e.g. "a photo of setup by banana", "a photo of skiing at

[1]https://hazyresearch.stanford.edu/blog/2022-04-02-domino

Listing 2: Templates used to generate captions for DOMINO (BERT).

```
    a photo of {LABEL} and [MASK]
    a photo of {LABEL} in [MASK]
    a photo of [MASK] {LABEL}
    [MASK] {LABEL} [MASK]
```

sandal") or did not refer to the target category at all (e.g. "three oranges and an apple on a white background" or "a photo of promoter david lemon" when the target is "lemon"). For reference, we include the full list of DOMINO (BERT) descriptions for ViT-H/14 in Listing 6, ResNet-50 in Listing 7, and the list of DOMINO (OFA) descriptions for ViT-H/14 in Listing 8 and ResNet-50 in Listing 9. In these lists, we prepend an asterisk in front of coherent topics, and we append the target category for each slice in parentheses. Of the starred coherent descriptions, we excluded three from evaluation because we could not find any related images in LAION-5B: "a photo of munitions and broom" (DOMINO (BERT), ViT-H/14), "a photo of primate and broom" (DOMINO (BERT), ResNet-50), and "a large banana sitting in the middle of an abandoned building" (DOMINO (OFA), both ViT-H/14 and ResNet-50).

**ADAVISION topics.** We also include the ADAVISION topics we compare to in Listing 3. Of the six categories {banana, broom, candle, lemon, sandal, wine bottle}, we recruited one user per category to test ViT-H/14 and generate topics, except for the categories *candle* and *wine bottle*, which one of the authors tested in a separate session. All users (including the author) were limited to 20 minutes for testing, and they had not explored ViT-H/14 on the tested class before.

## D. Additional details for finetuning

In Section 4.4, we presented results from experiments that show we can patch model performance on bugs while maintaining or slightly improving accuracy on the original ImageNet distribution, control topics, and OOD evaluation sets. In these experiments, we finetuned on 20 images from each of 30 buggy topics (which we call the *treatment topics*). These treatment topics are the same as in Section 4.3 / Appendix C, listed in Listing 3. In this section, we discuss hyperparameter choices, list all control topics, and break down OOD evaluation set gains by dataset.

**Finetuning hyperparameters.** We finetune ViT-H/14 using a small, constant learning rate of 1e-5 with the AdamW optimizer [2, 3] for five steps, with weight decay 0.01, batch size 16, and random square cropping for data augmentation. These hyperparameters were chosen in early experiments because they only degrade in-distribution model performance slightly. We report all results averaged over 3 random seeds along with the corresponding standard deviations. When deduplicating evaluation data against the finetuning data, we mark pairs as duplicates if their CLIP cosine similarly $> 0.95$.

Listing 3: Topics from ADAVISION studied in Sections 4.3 and 4.4 in the main text.

```
banana next to a banana smoothie
banana on kitchen countertop
banana in wooden woven basket
banana next to banana bread
toy banana
broom by fireplace
witch flying on a broom
photo of a person holding a boxy broom
silhouette of a person flying on a broom
broom in closet
black-and-white clipart of a candle
creamy white candle in glass jar
christmas candle next to tea
candle by window in snowstorm
person holding a candle at a vigil
grating a lemon
cooking with lemon
lemon tea with lemon
lemon on pancake with condensed milk or honey
lemon clipart
a lot of toy sandals
flip flop door wreath
translucent sandals
colorful flip flops
sandal ornament in a tree
top of a champagne bottle
wine bottle in a wiry wine rack
champagne in a champagne holder
wine bottle in a suit case
wine bottle with a wine stopper
```

**Control topics.** We provide a list of the 19 control topics from Section 4.4 in Listing 4. These topics were selected because they were semantically related to the treatment topics in Listing 3, but had different labels. For example, the ADAVISION topic `person holding a boxy broom` is visually similar to the concept of "person holding a mop", so we include the latter as a control topic. Other control topics are classes that were incorrectly predicted for a topic (*e.g.*, `banana on kitchen countertop` is frequently predicted "microwave", so we include "microwave in kitchen" as a control topic). We checked performance on these topics to make sure performance gains were

Listing 4: Control topics in Section 4.4 in the main text. In parentheses, we list the topic found by AdaVision with which the given control topic contrasts.

```
shopping basket (banana in wooden woven basket)
bread (banana next to banana bread)
dishwasher in kitchen (banana on kitchen
    countertop)
microwave in kitchen (banana on kitchen
    countertop)
eggnog (banana next to a banana smoothie,
    lemon on pancake with condensed milk or
    honey)
witch with cauldron (witch flying on a broom)
fireplace no broom (broom by fireplace)
mop (broom in closet,
    photo of a person holding a boxy broom)
person holding mop (photo of a person holding
    a boxy broom)
consomme (lemon tea with lemon)
black tea no lemon (lemon tea with lemon)
grated orange (grating a lemon)
pancake with condensed milk or honey no lemon
    (lemon on pancake with condensed milk or
    honey)
torch (person holding a candle at a vigil,
    candle by window in snowstorm)
clog shoe (sandal)
beer bottle (top of a champagne bottle)
suit case (wine bottle in a suit case)
empty wine rack (wine bottle in a wiry wine
    rack)
empty wire rack (wine bottle in a wiry wine
    rack)
```

not due to the model forming new shortcuts.

**Per-OOD evaluation set breakdown.** In Tables 4 and 5, we provide a breakdown of the OOD evaluation set performances; these were aggregated as an average in Table 2 of the main text. Table 4 displays the accuracy on treatment classes in each of the OOD evaluation sets, and Table 5 displays the overall accuracy in each of the OOD sets.

**Effect on conceptually unrelated bugs.** We evaluated whether finetuning on treatment topics affected conceptually unrelated bugs. For each class in {banana, broom, candle, lemon, sandal, wine bottle}, we found two additional topics with high failure rates (Listing 5), disjoint from the treatment topics in Listing 3. We then measured performance on these unrelated topics before and after finetuning, and we compare to performance changes on the treatment and control topics in Table 6. We see that finetuning on treatment topic improves performance on semantically unrelated bugs within the same set of classes, but gains are smaller than on treatment topics.

Listing 5: Topics from AdaVision studied in Sections 4.3 and 4.4 in the main text.

```
bananagrams
banana in fruit salad
curling broom on the ice
broomball with brooms
candle in mason jar with flower
hexagon candle
a lemon with a smiley face drawn on it
lemon on waffle
lace sandal
crocs
rows of wine bottles in a store
wine bottle in a wine fridge
```

| **Model** | **ImageNet** | **ImageNet V2** | **ImageNet-Sketch** | **ImageNet-R** | **ImageNet-A** | **ObjectNet** | **Avg. OOD** |
|---|---|---|---|---|---|---|---|
| Before finetuning | 87.7 | 65.0 | 86.1 | 89.2 | 65.6 | 84.3 | 78.0 |
| Finetuning with baseline | **93.1 (0.2)** | **71.7 (1.4)** | **90.9 (0.3)** | 90.5 (0.3) | 71.2 (1.3) | 86.4 (0.1) | 82.1 (0.6) |
| Finetuning with AdaVision | **92.9 (0.4)** | **69.4 (0.8)** | **91.7 (0.5)** | **93.9 (0.2)** | **76.8 (1.5)** | **87.9 (0.2)** | **84.0 (0.2)** |

Table 4: Accuracies on treatment classes, before and after finetuning. Results are averaged over three random seeds.

| **Model** | **ImageNet** | **ImageNet V2** | **ImageNet-Sketch** | **ImageNet-R** | **ImageNet-A** | **ObjectNet** | **Avg. OOD** |
|---|---|---|---|---|---|---|---|
| Before finetuning | 88.4 | 81.0 | 64.4 | 89.1 | 83.9 | 69.9 | 77.7 |
| Finetuning with baseline | **88.5 (0.0)** | **81.3 (0.0)** | 64.5 (0.0) | 89.2 (0.1) | 84.4 (0.1) | **70.5 (0.2)** | 78.0 (0.1) |
| Finetuning with AdaVision | 88.4 (0.0) | 80.9 (0.1) | **64.7 (0.0)** | **90.0 (0.0)** | **84.9 (0.0)** | **70.5 (0.0)** | **78.2 (0.0)** |

Table 5: Accuracies on all classes, before and after finetuning. Results are averaged over three random seeds.

| **Model** | **Treatment Topics** | **Control Topics** | **Unrelated Topics** |
|---|---|---|---|
| Before finetuning | 72.6 | 91.3 | 61.0 |
| Finetuning with baseline | 82.5 (0.9) | 90.8 (0.3) | 65.6 (0.8) |
| Finetuning with AdaVision | **91.2 (0.5)** | **91.2 (0.2)** | **74.7 (2.0)** |

Table 6: Accuracies on treatment, control, and unrelated topics. Finetuning on treatment topic improves performance on semantically unrelated bugs within the same set of classes, but gains are smaller than on treatment topics.

Listing 6: Slice descriptions generated by Domino (BERT) for target model ViT-H/14. Asterisks in front of coherent topics.

```
a photo of setup by banana (banana)
*a photo of wine and banana (banana)
*a photo of ceramics and banana (banana)
*a photo of basket and banana (banana)
*a photo of munitions and broom (broom)
a photo of activist david broom (broom)
a photo of synthesizer. broom (broom)
*a photo of wildlife at broom (broom)
a photo of violinist jenny broom (broom)
*a photo of literature and candle (candle)
a photo of panchayats candle (candle)
*a photo of altarpiece and candle (candle)
a photo of rob and candle (candle)
a photo of corella lemon (lemon)
*a photo of blender lemon (lemon)
a photo of promoter david lemon (lemon)
a photo of clown billy lemon (lemon)
a photo of estadio jose lemon (lemon)
a photo of rowing on sandal (sandal)
a photo of nana and sandal (sandal)
a photo of placental sandal (sandal)
a photo of skiing at sandal (sandal)
a photo of screenwriter michael sandal (sandal)
a photo of shelter and wine bottle (wine bottle)
*a photo of champange wine bottle (wine bottle)
*a photo of bakery and wine bottle (wine bottle)
*a photo of advertisement on wine bottle (wine bottle)
*a photo of grocery stores wine bottle (wine bottle)
```

Listing 7: Slice descriptions generated by Domino (BERT) for target model ResNet-50. Asterisks in front of coherent topics.

```
*a photo of ceramics and banana (banana)
a photo of reception by banana (banana)
*a photo of orange and banana (banana)
a photo of neutron star banana (banana)
a photo of architect paul banana (banana)
a photo of singer jenny broom (broom)
a photo of rowing on broom (broom)
*a photo of ornate old broom (broom)
*a photo of factory of broom (broom)
*a photo of primate and broom (broom)
*a photo of blowing the candle (candle)
a photo of consultant john candle (candle)
*a photo of colorful birds candle (candle)
a photo of swamy candle (candle)
*a photo of candle in entryway (candle)
red lemonade series. (lemon)
liz lemon and the observer (lemon)
keith lemon and david bowie (lemon)
liz lemon and the batman (lemon)
a photo of lemon bay shuttle (lemon)
a photo of cognitive development sandal (sandal)
a photo of drilling the sandal (sandal)
a photo of lecture at sandal (sandal)
*a photo of frozen black sandal (sandal)
a photo of longevity by sandal (sandal)
*a photo of golden wine bottle (wine bottle)
a photo of autopsy wine bottle (wine bottle)
*a photo of home and wine bottle (wine bottle)
a photo of libertarian wine bottle (wine bottle)
a photo of estadio wine bottle (wine bottle)
```

Listing 8: Slice descriptions generated by Domino (OFA) for target model ViT-H/14. Asterisks in front of coherent topics.

```
a plate on a table with knives and forks (banana)
*a banana next to a bottle of wine and a glass (banana)
*a kitchen counter with bananas and a pineapple on it (banana)
*a banana and two pears in a red basket (banana)
*a large banana sitting in the middle of an abandoned building (banana)
three mops and a bucket against a brick wall (broom)
a man holding a baseball bat in a room (broom)
a woman in a blue dress is looking at a computer (broom)
a green praying mantis standing on a piece of wood (broom)
*a woman sitting on a chair holding a broom (broom)
*a carved pumpkin with a candle in the middle (candle)
*a candle sitting on the ground on a brick floor (candle)
*a group of lit candles in front of a stained glass window (candle)
*a man sitting at a table with candles (candle)
a white bird perched on a tree branch eating (lemon)
*a pitcher pouring lemonade into a glass with lemons (lemon)
three oranges and an apple on a white background (lemon)
a display of tomatoes and other vegetables (lemon)
a bunch of oranges sitting on top of a table (lemon)
a pair of shoes sitting on top of a skateboard (sandal)
a woman holding a small child on her lap (sandal)
*a pink crocheted sandal with a flower on it (sandal)
*a person is wearing a black sandal on their foot (sandal)
a woman laying on a bed with a laptop (sandal)
a group of small bottles of liquor on a table (wine bottle)
*a bottle of champagne in a bowl on a table (wine bottle)
*a bottle of wine and a paper on a counter (wine bottle)
*a glass of wine and a bottle on a table (wine bottle)
*a bottle of wine and a cigar on a table (wine bottle)
```

Listing 9: Slice descriptions generated by Domino (OFA) for target model ResNet-50. Asterisks in front of coherent topics.

```
*a green bowl with some bananas and a piece of fruit (banana)
a plate on a table with knives and forks (banana)
*a bowl of oranges and bananas on a table (banana)
*a banana and two pears in a red basket (banana)
*a large banana sitting in the middle of an abandoned building (banana)
*a broom sitting on the floor in front of a wooden door (broom)
a group of people holding a large wooden stick (broom)
*a close up of a broom with a wooden handle (broom)
three mops and a bucket against a brick wall (broom)
*a broom hanging on the side of a porch (broom)
*a baby girl sitting in front of a birthday cake with a candle (candle)
*a little girl is holding a candle and looking up (candle)
*a group of lit candles in front of a stained glass window (candle)
*a candle sitting on the ground on a brick floor (candle)
*a candle sitting on top of a wooden table (candle)
a group of sliced oranges and kiwi fruit (lemon)
*a pitcher pouring juice into a glass with lemons (lemon)
three oranges and an apple on a white background (lemon)
*a lemon with a smiley face drawn on it (lemon)
*a bowl filled with oranges and a lemon (lemon)
a pair of snoopy shoes and a box on a green table (sandal)
*a woman is wearing a pair of sandals on her feet (sandal)
*a woman wearing sandals standing on a concrete floor (sandal)
a pair of shoes sitting on top of a magazine (sandal)
*two pictures of a woman wearing a pair of sandals (sandal)
*a vase of roses and two bottles of wine (wine bottle)
*a cake with a bottle of wine in a box (wine bottle)
*a bottle of wine and grapes on a counter with a glass (wine bottle)
*a woman next to a row of wine bottles (wine bottle)
*a bottle of wine and a cigar on a table (wine bottle)
```

## References

[1] Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022. 7

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. https://openreview.net/forum?id=Bkg6RiCqY7. 8

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1912.01703. 8

[4] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400, 2019. 3

[5] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR, 2021. 2