6 Appendix

Our code is available at https://anonymous.4open.science/r/HiBug-0EE7.

6.1 Model repair by HiBug

We elaborate on the details of data selection methods and data generation methods of *HiBug*.

Data selection. Our data selection method aims to identify crucial data for model repair from the unlabeled data pool. To achieve effective repair, we require the selected samples to exhibit both high error rates and diversity.

To accomplish this objective, our first step is to identify data slices with high error rates. Given a predetermined budget and an unlabeled data pool, we assign each data point in the unlabeled pool to its respective data slice based on attribute values. However, a challenge arises because the labels for these data points are unavailable. Therefore, we turn to the validation set, which already possesses known labels. By leveraging the labels in the validation set, we can calculate the failures and the error rate associated with each data slice. This information allows us to estimate the error rates of the unlabeled data slices.

Once we have obtained the error rate for each slice, we rank all data slices according to their error rates and select those with high error rates. This process is carried out iteratively, starting with the data slice that exhibits the highest error rate. We accumulate the number of failures associated with the selected slices as the iterations progress. The iteration continues until the accumulated number of failures equals or exceeds a threshold defined by the hyper-parameter β multiplied by D_F. Here, β represents the percentage of failures we aim to address, while D_F corresponds to the total number of failures observed in the validation set.

After identifying slices with high error rates, we distribute the budget among the data slices proportionally to the number of failures they encompass. This allocation ensures that data slices with a higher number of failures receive a larger share of the budget, while also considering the diversity factor. Finally, we select unlabeled data from each data slice based on the budget allocated to that particular slice. During this selection process, we prioritize data points with lower confidence to maximize the potential for model improvement. In our experiments, we set the hyper-parameter β to a fixed value of 0.9.

Data generation. *HiBug* can generate more related data (e.g., data with failure patterns) that can further improve the model performance. For data generation, we employ a straightforward approach. Firstly, we collect the attribute values of the top 200 data slices with the highest validation error rates for each label. Next, we construct a prompt that incorporates label name and attribute variables:

$$\textit{"A photo of *1 *2 *label *3, in a *4 background"} \tag{2}$$

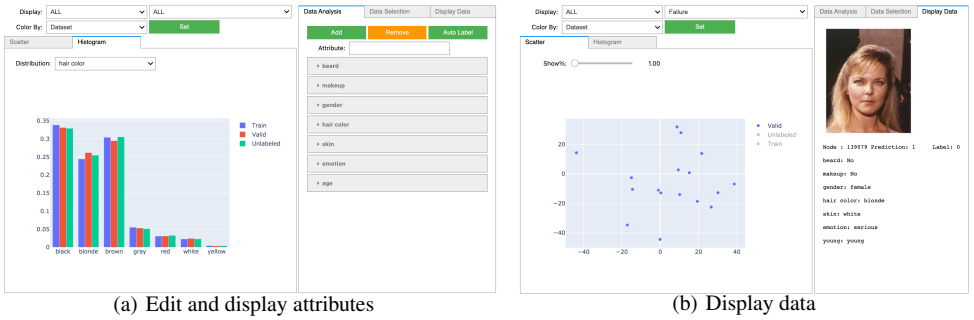
During the data generation process, the variables *1, *2, *3, *4 are replaced by the values of attributes, such as "several" "black" "with a person" and "snow". The variable *label is substituted with the corresponding label name, such as "dog" or "cat."

We show the effectiveness of data generation with *HiBug* in ImageNet10. In this experiment, we generate a total of 10,000 data points, with 1,000 data points generated for each label. We downsample the overall generated data to obtain results for 1,000 data points and 5,000 data points, as presented in Table 5. For comparison, we use a baseline method "Class name" that uses the following prompt, where the variable *label will be substituted with the corresponding label name during generation.

$$\textit{"A photo of *label."} \tag{3}$$

6.2 Introduction to the user interface

We offer a user interface for *HiBug* to facilitate more effective debugging. The user interface provides two primary functionalities. Firstly, users can easily view the distribution of attributes. For each attribute name, a histogram representing the various attribute values can be plotted, as demonstrated



(a) Edit and display attributes (b) Display data

Figure 6: In (a), we show that users can view the distribution of attributes by histogram (left tab), and propose or edit attributes (right tab). In (b), we show that users can view the distribution of features by scatter plot (left tab), and attributes of a specific data point in the scatter (right tab).

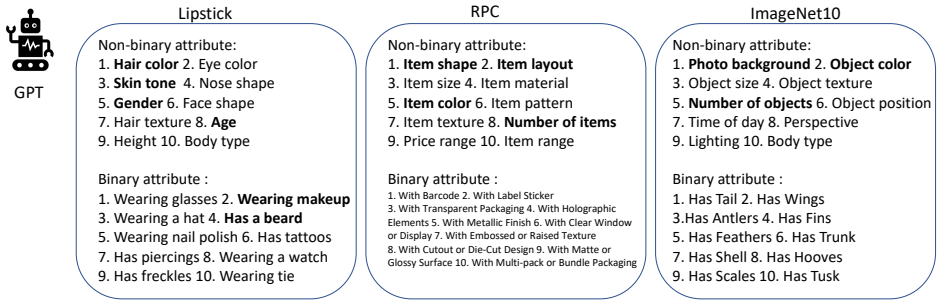


Figure 7: Attribute names proposed by chatGPT. The value of binary attributes are "yes" or "no".

in Figure 6 (a). It's worth noting that we also provide additional flexibility by allowing users to input attribute names during the attribute proposal stage. This feature can be particularly useful when users are confident about specific tasks and seek to enhance the debugging process. Secondly, users can examine the data distributions in the embedding space. For example, we can display and analyze specific failures, as depicted in Figure 6 (b).

6.3 Experiment details

chatGPT attribute proposal. Figure 7 shows the attribute names proposed by chatGPT in the three experiment settings discussed in the main paper. We also provide additional flexibility by allowing users to select attribute names for HiBug to explore. We highlight the attributes selected for our experiments.

HiBug 's bug discovery outputs. Figure 8 showcases the bug discovery outputs obtained by HiBug in the three experiment settings outlined in the main paper. These outputs demonstrate the effectiveness of HiBug in identifying bugs and providing valuable insights to the user.

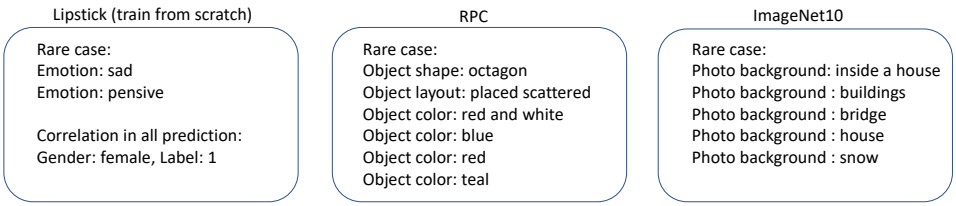


Figure 8: HiBug 's bug discovery outputs on three experiment settings.

Table 6: Given the same attribute list for each data, HiBug can discover more error slices with a larger validation set.

Size of validation set	15k	10k	5k	1k	0.5k
Number of error slices	288	238	168	71	49

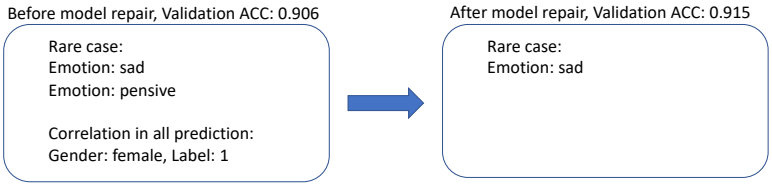


Figure 9: HiBug 's outputs before and after repair.

6.4 Further ablation studies

Size of the validation set. We conducted additional ablation studies to explore the impact of the validation set size on HiBug 's performance. To quantify the error slices, we define them as data slices with validation performance lower than or equal to the overall validation performance. As presented in Table 6, we observed that HiBug can discover a greater number of error slices when the validation set size is larger. This phenomenon can be attributed to the larger dataset covering a broader range of attribute values, leading to the identification of a higher number of error slices by HiBug .

Do bugs fixed after model repair? In the lipstick experiment, the model's predictions are correlated with the gender of the person in the data. This correlation is detected in the bug discovery process of *HiBug*. For comparison, we use HiBug to test the model after repairing by data selection. The result is presented in Figure 9. We observe that most bugs discovered before repair disappear. This confirms again that our model repair strategy is effective.