Identifying and Mitigating Model Failures through Few-shot CLIP-aided Diffusion Generation

Anonymous authors
Paper under double-blind review

Abstract

Deep learning models can encounter unexpected failures, especially when dealing with challenging sub-populations. One common reason for these failures is the occurrence of objects in backgrounds that are rarely seen during training. To gain a better understanding of these failure modes, human-interpretable descriptions are crucial for further analysis and improvement which is expensive. In this study, we propose an end-to-end framework that utilizes the capabilities of large language models (ChatGPT) and vision-language deep models (CLIP) to generate text descriptions of failure modes associated with spurious correlations (e.g. rarely seen backgrounds) without human-in-the-loop intervention. These descriptions can be used to generate synthetic data using generative models, such as diffusion models. The model can now use this generated data to learn from its weaknesses and enhance its performance on backgrounds that are uncommon for each class of data. Our approach serves as a broad solution, promising progress in comprehending model failure modes and strengthening deep learning models across a wide range of failure scenarios (e.g. bacckgrounds, colors) automatically in a few-shot manner. Our experiments have shown remarkable **improvements in accuracy ($\sim$ 21%)** on hard sub-populations (particularly for wrong background association) across 40 different models, such as ResNets, EfficientNets, DenseNets, Vision Transformer (ViT), SwAVs, MoCos, DINOs, and CLIPs on various datasets such as ImageNet-1000, CIFAR-10, and CIFAR-100.

1 Introduction

The quality of training data directly impacts the performance and robustness of machine learning models. Despite careful curation of training data, models can still exhibit failure modes where their performance deteriorates in specific sub-populations of data, leading to misclassifications or inaccurate predictions Jiang et al. (2018); Arpit et al. (2017). The failure modes of deep networks can arise from various factors, such as noisy labels Sukhbaatar et al. (2014); Jiang et al. (2018); Reed et al. (2015), multi-labels Zhang et al. (2018b), and spurious correlations Zhou et al. (2020), particularly when it comes to distinguishing between objects and their backgrounds Kattakinda & Feizi (2021); Xiao et al. (2020). (See the figure 7 in appendix for examples of these failures.)

Similar to how humans use image backgrounds as cues for object recognition, studies have shown that machine learning models also rely on backgrounds when making decisions. In some cases, models may prioritize backgrounds to the point of overlooking important object features for classification Zhang et al. (2007); Ribeiro et al. (2016); Sagawa et al. (2019).

Various approaches have been attempted to address failure modes caused by spurious background associations, but many of them are inadequate in addressing all aspects of the problem. Some approaches involve human-in-the-loop interventions Mitchell et al. (2021); Santurkar et al. (2021), which is labor-intensive and challenging to apply to large-scale operations. Moreover, many of them only target a limited set of failure modes, neglecting the comprehensive spectrum of potential failures Barbu et al. (2019); Hendrycks et al. (2021a); Hendrycks & Dietterich (2019); Hendrycks et al. (2021b); Kattakinda & Feizi (2021). Additionally,

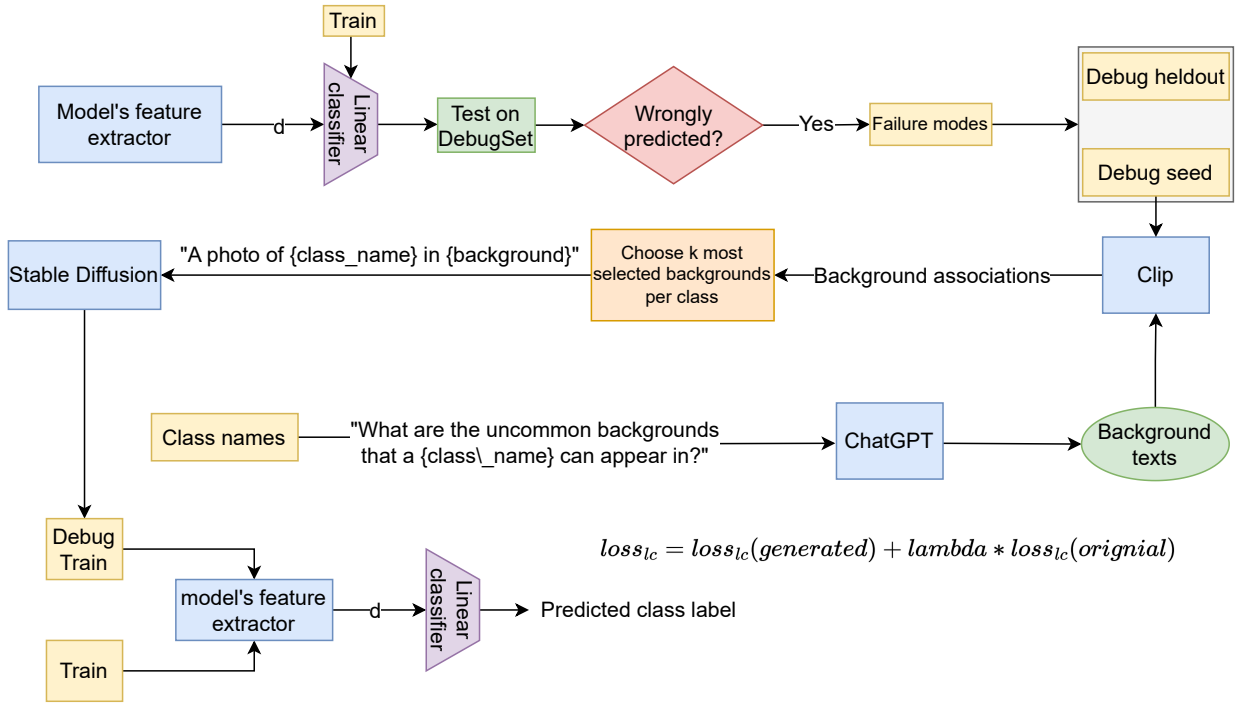


Figure 1: A summary of our approach: For a model based on the wrongly predicted debug samples (failure samples), we form two sets - debug seed and debug heldout. We use the debug seed set to address the model's failures by inputting them to CLIP Radford et al. (2021), along with a set of backgrounds obtained from ChatGPT where objects are less likely to occur. We then obtain a set of backgrounds and remove redundancies, and generate synthetic data by inputting the prompt "A photo of {class_name} {background}" to Stable Diffusion Rombach et al. (2022). With this synthetic data that precisely captures the model's failure modes, we can now debug the model's predictions on other test data by training a very low-cost network on top of our model, which assigns different weights to the original data and the generated data.

some of the existing work lacks clear descriptions of model failures in a human-understandable manner, posing challenges in terms of interpretability and validation.

Alongside research on identifying failure modes, there are several debugging approaches that seek to utilize these failure modes to enhance the accuracy of machine learning models. These include creating supplementary datasets with failure samples to help the model learn robust features Xiao et al. (2020); Singla et al. (2022), or modifying the model's parameters to incorporate information from identified failure modes Rame et al. (2022). These studies lack failure mode descriptions that are easily understandable for humans, making it challenging to interpret their validity.

2 Our contribution

This research leverages recent generative models, large language models, and CLIP to introduce an automated framework addressing failure modes (spurious correlations) in diverse task-specific deep learning models. The framework, outlined in Figure 1, answers key questions such as identifying and rectifying spurious associations leading to model failure, utilizing these failure modes to enhance model performance, exploring patterns in failure modes across a model group, and using a single set of auxiliary data to simultaneously improve a subgroup of models.

To summarize, our approach initially identifies all model failures on a specific subset, denoted as ***DebugSet***, within the validation set. We then pinpoint spurious correlations, such as background associations, for each dataset class by querying ChatGPT with "What are the uncommon backgrounds that a class_name can appear in?" and remove redundacies after obtaining all uncommon backgrounds. Subsequently, a zero-shot classification using CLIP identifies the background for each failure among all the uncommon backgrounds. To enhance model performance, we generate k artificial images with prompts like "[class_name] in [background_name]" and incorporate this supplementary data into the original training set. In the second phase, we demonstrate that models with similar architectures exhibit analogous failures, allowing for efficient troubleshooting of a model group using a single set of generated auxiliary data. This approach proves both time and memory efficient. The results of our experiments, detailed in section 5, underscore the effectiveness of this straightforward method in achieving interpretability and debugging goals.

Our paper presents several contributions to the field of machine learning model failure analysis and debugging. These contributions include, but are not limited to, the following:

- **Generalizability**: Introducing a comprehensive end-to-end framework that interprets and rectifies failures arising from specific spurious associations, such as incorrect background, color, and size correlations, which can contribute to any model inaccuracies.
- **Failure Inspection**: Identification of spurious background association failure modes of ~ 40 multiple models on ImageNet in an interpretable manner (section 5.2.1), and exploring common patterns in failure modes among individual models with same architectures (section 5.2.2) automatically and without any human intervention.
- **Failure Mitigation**: Improving the performance of individual models on challenging subpopulations (5.3.1), and boosting the performance of model subsets by employing a unified set of auxiliary data, leveraging shared failures to enhance efficiency in both time and memory usage (section 5.3.2).
- **Collective Failure Mitigation**: Enhancing subsets of models' performance through a unified set of auxiliary data owing to their shared failures which saves time and memory. To the best of our knowledge, this work represents the first effort to collectively address failures within a subgroup of models simultaneously. (section 5.3.2).
- **Dataset creation for debugging ImageNet, Cifar10, and Cifar100**: We created a dataset that will improve models' performance on background and color associated failure modes on these three datasets.

3 Related work

3.1 Failure mode detection

Numerous studies have been conducted to detect machine learning model failure modes. As previously mentioned, some involve human-in-the-loop methods, where failure examples are reviewed to identify common patterns Mitchell et al. (2021); Santurkar et al. (2021). Others adopt automated approaches by introducing frameworks that effectively capture model failures Chung et al. (2019); Singla et al. (2021); Nushi et al. (2018); Singla & Feizi (2021); Wong et al. (2021); Wu et al. (2019); Zhang et al. (2018a); Jain et al. (2022). For instance, Chung et al. (2019) employs a technique that slices the validation data to isolate sections where the model performs poorly. Singla et al. (2021) identifies visual attributes that lead to inadequate performance when present or absent. Jain et al. (2022) identifies and represents model failures as directions in the latent space, and Eyuboglu et al. (2022) that proposes an evaluation framework to systematically compare (slice discovery method) SDMs across diverse slice settings by generating captions for hard subpopulations. Distinguishing itself from existing methodologies, our approach provides enhanced generality by **permitting the explicit selection of the spurious correlation targeted for mitigation**. For instance, although the approach presented by Kattakinda et al. Kattakinda et al. (2022) effectively tackles spurious correlations tied to foreground and background features by learning disentangled representations,

it encounters difficulties when confronted with a wider spectrum of spurious correlations, e.g. color. This is due to the inherent challenge of learning disentangled representations for many spurious correlations in isolation from the foreground object.

3.2 mitigation of hard subpopulations and interpretability of models

Several methodologies leverage extracted failure modes to debug and enhance the performance of deep learning models. Singla et al. Singla et al. (2022) introduce a framework that identifies visually similar images to model failures and incorporates them as new data for debugging various models. Kattakinda et al. Kattakinda et al. (2022) focus on learning invariant features for foreground and background by penalizing the mutual information between the features and background/foreground labels. This approach contributes to robust model training, particularly by addressing issues related to spurious correlations.

In the context of data generation, Bansal and Grover Bansal & Grover (2023) and Wiles et al. Wiles et al. (2022) use generated data to diversify training datasets. However, it's essential to note that their methods do not specifically target failure modes like spurious correlations. They rely on class names and general captions for generating auxiliary data, which may not be tailored to address specific failure modes.

Moreover, Wiles et al. Wiles et al. (2022) propose a bug discovery approach using off-the-shelf image generation and captioning, contributing to the interpretability of model predictions. On the other hand, Jain et al. Jain et al. (2022) leverage Support Vector Machines (SVMs) to distill model failures as directions in latent space, focusing on latent representations of model failures.

In comparison to existing methodologies that address failure modes on specific datasets, our framework introduces two noteworthy contributions. Firstly, **it achieves enhanced model performance with significantly fewer generated examples, (5 for each failure)**. Secondly, **our experiments extend to collective debugging, demonstrating the ability to improve a subset of model failures by generating a single auxiliary artificial dataset based on only one model's failures**. This is particularly valuable given our observation that models within the same categories exhibit similar failures, a phenomenon also noted in Wiles et al. (2022).

Moreover, our approach stands out for its efficiency, eliminating the necessity for complete model retraining or fine-tuning. We exclusively focus on retraining the linear head for classification, streamlining the failure mode mitigation process.

4 Main method

4.1 Failure-mode detection

A common reason for accuracy drops during inference is the model's learned spurious correlations from training. For example, Associating objects with backgrounds, a spurious correlation, can hinder the model's ability to learn objects themselves. This challenge arises when the model encounters objects in unfamiliar backgrounds during testing, notably in computer vision tasks where backgrounds define object context. To tackle this, introducing the model to a range of scenarios that address the particular failure mode (such as color or background associations) we aim to mitigate, can improve its ability to identify objects in different contexts, and avoid correlating the objects and their changable features (e.g. color) or contexts (e.g. background).

In this work we explain how we use this framework for wrong background associations, however it can easily be applied to all other spurious correlations that models may mistakenly learn.

To address and rectify failures attributed to backgrounds, we utilize the feature extractor for each model on the datasets, generating a feature vector for each data point. The subsequent linear head atop this feature extractor is responsible for executing the classification task. Instances where the model makes incorrect predictions form a set termed the ***"debug set"***. This debug set serves as a tool for identifying and resolving failure modes, as it comprises all examples where the model fails. While these failures may stem from various

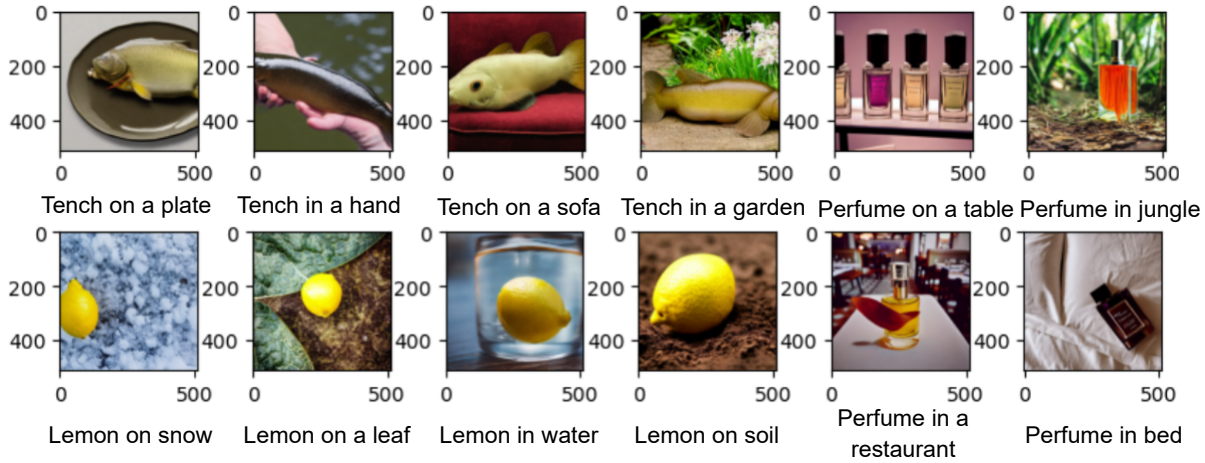


Figure 2: Examples of generated data by Stable Diffusion

factors, our experiments underscore the significance of mitigating incorrect background associations, as it significantly improves the performance of all models.

Class name	Uncommon backgrounds
Sea lion	Desert, Rain forests, Urban Areas, Polar Ice Caps, Caves, Grasslands, Volcanic Areas
Siberian husky	Jungle Canopies, In sky, Caves, Underwater, Indoor Spaces, Marshlands, Tropical Rainforests
croquet ball	Mountain Peaks, Busy Streets, Frozen Lakes, Underneath Building Foundations, Subway Tunnels, in a restaurant
lipstick, lip rouge	Gyms and Fitness Centers, Swimming Pools, Medical Facilities, Construction Sites, Sports Events, Military Training

Table 1: Examples of suggested uncommon backgrounds for a class of data by ChatGPT

4.2 Failure-mode textualization

Vision-language models are popular as they can provide more comprehensive understanding of complex phenomena by combining information from different modalities like text, images, and audio, enabling them to interpret data in a more human-readable form Lu et al. (2019); Chen et al. (2018); Mithun et al. (2020).

Understanding failure modes is critical for validating proposed debugging methods. By identifying the causes of failure, we can improve our models and refine our data collection methods. For each class_name in our dataset, we first prompt ChatGPT "What are the uncommon backgrounds that a class_name can appear in?" and filter out the redundant suggested backgrounds. Some examples can be seen in Table 1. Then, we use CLIP Radford et al. (2021) to interpret failure modes by splitting the failures from the ***debug set*** into two sets called ***debug seed*** and ***debug heldout***. We then perform zero-shot classification by passing ***debug seed*** along the set of class_wise uncommon backgrounds proposed by ChatGPT to a CLIP model, so for each data point, CLIP will opt for the background that is more likely to be the actual background of the object shown in the image. For each data class, we then pinpoint the $\boldsymbol{k}$ most frequently selected backgrounds by CLIP, which the model failed to classify. This will provide valuable insights into the ways how a model may fail when confronted with a particular selected spurious association.

Figure 3: Some examples of failure modes of ResNets and DINOs

4.3 Generating synthetic data

By leveraging the detected backgrounds of failures by Clip, we can both interpret those failures, and use them to debug models. For instance, in the case of the Imagenet class "tench," errors predominantly occur when the fish is held by a person's hand, a scenario rarely encountered during training. To address this, a generative model like Stable Diffusion Ho et al. (2021) can be used to create images that familiarize the model with diverse object contexts. For the "tench" class, we generate data by inputting the prompt "tench in a hand" to the Stable Diffusion. Examples of such generated data are presented in figure 2.

4.4 Retraining the linear head

After collecting the additional synthetic data for the failed scenarios, which we call ***debug_train***, we can use it along with our original trainset to debug our models. To achieve this, we only need to train a linear head on top of the model's feature extractor for the classification purpose and not the whole model. It is important to note that we assign different weights to the datapoints from the original_train set and the Debug_train set in our linear head training loss 1. This parameter is called ***lambda***, and in our experiments, we report its effect on the overall performance of the model. By incorporating the additional debug_train data and carefully tuning the lambda parameter, we can potentially improve the performance of our models.

$$L_{cl} = L_{cl}(Original_train) + \lambda * L_{cl}(Debug_train) \tag{1}$$

5 Experiments

5.1 Setting

We load our datasets and use their training data to train the linear head on top of the models' extracted features. For ImageNet, we use 30 data points per class and the overall 30000 training images. We choose 30000 images out of 50000 of Imagenet's validation set as our *debug_set*, and the remaining 20000 samples will be considered for the testing process. Each image's resolution is $224 * 224$, and the task performed here

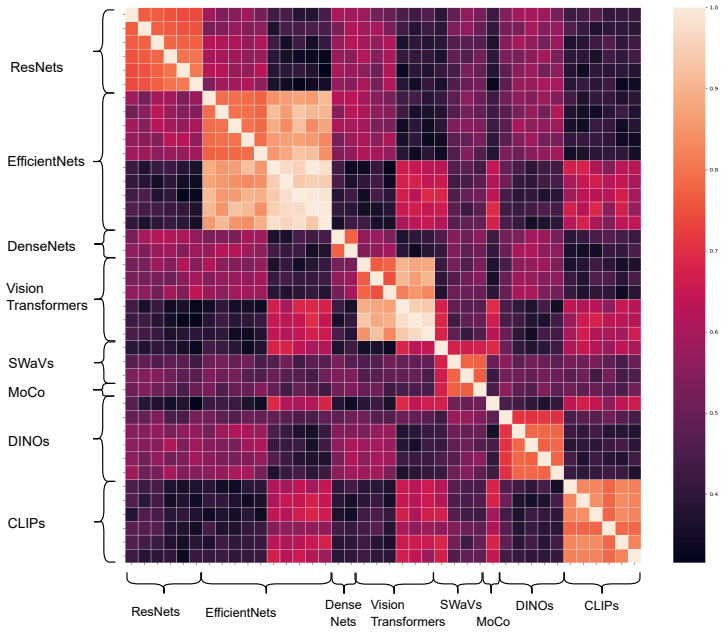


Figure 4: Comparing failures of all models **(Intersection/Union)**.We observe that models belonging to the same categories tend to exhibit more comparable failures.

is the classification task on Imagenet classes. Other settings include hyper-parameters that we used can be seen in table 4 in the appendix.

We tested our method on 40 different models including ResNets He et al. (2016), EfficientNets Tan & Le (2019), DenseNets Huang et al. (2017), Vision Transformer (ViT) Dosovitskiy et al. (2021), SwAVs Caron et al. (2020), MoCos He et al. (2019), DINOs Caron et al. (2021), and Clips Radford et al. (2021). The exact list of used models can be seen in table 5 in the appendix. For the sake of space, we only show experiments on DINO and ResNet models, and experiments for other models can be found in appendix 6.

We split the detected failures of *debug set* in half. The first half, *debug seed* will be used for debugging. For obtaining uncommon backgrounds, we use ChatGPT 3.5. The CLIP model we use for choosing backgrounds for data points is ViT-B32 CLIP. For generating synthetic data, we use Stable Diffusion $V1-5$ imported from the diffusers package.

5.2 Failure inspection

The initial stage of our framework involves analyzing how various models fail to classify objects on different datasets. To accomplish this, we use the CLIP model to identify backgrounds on which models struggle to classify objects. This results in captions that describe failures related to rare backgrounds. In the following stage, we examine these identified failures and explore how the generated captions help us to recover from them. We investigate results for both **individual and Collective failure inspection**.

5.2.1 Individual Failure Inspection

In figure 3, we show some instances where ResNet and DINO models have failed and see that these failures are due to wrong background association. In this figure, the six images on the left (a-f) are examples of Resnets' failures, and the five images on the right (g-k) are failure modes' of DINO models. For example, image ***c*** shows *"a wallet in jungle"*, which can be regarded as an uncommon background for this object. As a result, the ResNet34 model is unable to classify it accurately and instead predicts a *"mushroom"* which is more likely to be found *"in garden"*, especially under a plant, despite having no resemblance to the actual

object in the image. Similarly, image $\boldsymbol{h}$ illustrates *"a bell or wind chime in sky"*, which is uncommon since *"bell"* is more likely to be seen with other backgrounds such as *"a door, a building or a wall"*. Therefore, the DINO_reset50 model mispredicts as *"traffic light"* because *"traffic light"* is more common to be seen *"in a sky background"*.

In general, our approach is capable of addressing failure scenarios originating from uncommon backgrounds of objects. Analyzing the relevant backgrounds allows us to readily understand the cause of such failure instances.

5.2.2 Collective Failure Inspection

Within this section, we will conduct a comparison of the failure modes for all models to assess their alignment. Our aim is to determine the extent to which failures are consistent across models. While numerous studies have focused on analyzing similarities in the learning process and representations of different models, such as Raghu et al. (2021) that demonstrated the similarity between convolutional neural networks (convnets) and other convnets, as well as the similarity between vision transformers (ViTs) and other ViTs, our focus is on investigating whether models also fail in similar ways. This will enable us to gain a deeper understanding of how to address the issue of failures in a more generalized manner without taking the specific model into consideration.

The failures of different models in various categories are compared in figure 4 by computing the *intersection over union* of the failures. It can be observed that models within the same category fail in more similar samples. Typically, the failures between models from the same category are over 80% similar (e.g. CLIPs and EfficientNets). Among all 40 models, the intersection of failure modes is above 40%, indicating that models tend to fail in very similar ways, even with different architectures. This will raise the question of "how to utilize this similarity in failures to enhance a group of models' performance?" which we will explore more in section 5.3.2.

It is pertinent to mention that Wiles et al. (2022) has also recognized patterns of consistencies in failures among models within the same category. however, we take a step further and delve into leveraging these consistencies to systematically mitigate shared failure modes.

5.3 Failure Mitigation

5.3.1 Individual Failure Mitigation

The outcomes from employing our framework are presented in Table 2. we only included the results for ResNets and DINOs, but we have results for other models (EfficientNets, DenseNets, ViTs, SWaVs, MoCo, and CLIPs) in the appendix. In Table 2, we constructed *debug_seed* and *debug_heldout* sets to yield zero accuracy for the model, as they are composed of model failures. Post debugging and utilizing *debug_seed*, we observe substantial improvements in *debug_heldout* data that we didn't use for debugging, ensuring an unbiased evaluation. This improvement underscores that many failure modes stem from incorrect associations models make between objects and backgrounds. Some might argue that this gain results from the additional data. Thus, we present results for a baseline we term **Random_debugging**. This baseline similarly uses *debug_seed* and *debug_heldout*, then generates synthetic data using only class names (prompts are structured as *"A photo of [class_name]"*). This comparison illustrates that the improvement of our method arises from considering background information. In the outcomes of Random_debugging, the improvement over debug_seed and debug_heldout is roughly equivalent since no information from the background association of either set was utilized. Random_debugging solely employs class names to generate data. However, this improvement is not on par with the gain achieved by incorporating background information when generating new data.

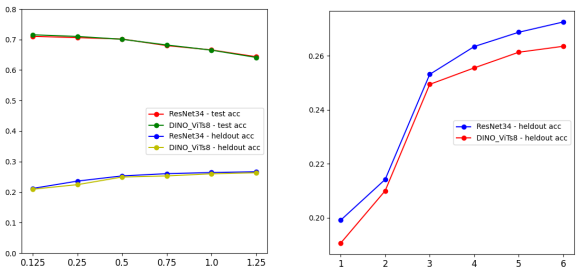
It's worth noting that despite incorporating Stable Diffusion-generated data, which could be seen as out-of-distribution samples, a positive impact on model performance remains. This is primarily attributed to the parameter λ that controls the contribution of the generated images in our training process. The influence of this parameter is depicted in figure 5.

Models		Accuracies						
Model category	model name	Accuracy before debugging	**Accuracy of Individual Debugging** (ours)			Accuracy of Random debugging		
		Test	Test	seed	**heldout**	Test	seed	heldout
ResNet	resnet18	0.6236	0.6413	0.2636	**0.2128**	0.6242	0.1134	0.1129
	resnet26	0.6593	0.6671	0.2856	**0.228**	0.6604	0.09539	0.0904
	resnet34	0.7017	0.7165	0.3061	**0.2531**	0.7099	0.09856	0.08615
	resnet50	0.7631	0.7671	0.3444	**0.2717**	0.7657	0.1102	0.1072
	resnet101	0.796	0.8024	0.3574	**0.2656**	0.7974	0.1132	0.1181
	resnet152	0.816	0.8238	0.3609	**0.2817**	0.8183	0.08207	0.08804
DINO	ViTs8	0.6977	0.7008	0.3117	**0.2494**	0.6979	0.1134	0.1129
	ViTs16	0.649	0.6558	0.2922	**0.2379**	0.6502	0.09539	0.0904
	ViTb8	0.7101	0.7136	0.3325	**0.2518**	0.7104	0.09856	0.08615
	ViTb16	0.6832	0.6851	0.3067	**0.2477**	0.681	0.1102	0.1072

Table 2: Accuracy of our method compared to the Random_debugging. Note that the accuracy of models on debug_seed and debug_heldout was zero before debugging. After applying our debugging method, we gain above $\sim 21\%$ improvements in accuracies for all models, showcasing that more than $\sim 21\%$ of model errors in the heldout set come from wrong background associations.

Another crucial hyper-parameter is the number (#) of generated synthetic data per class. The effect of this hyper-parameter, denoted as $\boldsymbol{k}$, is illustrated in figure 5.

The improvement observed in debug_heldout data surpasses $\sim 21\%$ for all models, highlighting the tendency of models to fail in associating backgrounds with objects and utilizing this association to predict objects, neglecting object-specific features. This can be contrasted with the accuracy gain achieved by the **Random_debugging** baseline, which is significantly smaller compared to our method.



(a) Effect of hyper parameter lambda on both test and heldout accuracies.

(b) Effect of hyper-parameter k (# cho--sen backgrounds per class) on heldout accuracy

Figure 5: a) As the value of *lambda* increases, the accuracy on the heldout set improves while the accuracy on the test set decreases. However, there is a specific point **(0.5)** on the plot where the accuracy of the models on both the test and heldout sets stabilizes. b) Increasing the number of chosen backgrounds per class enhances the accuracy on the heldout set. Considering the high cost of generating additional data, we opt for $\mathbf{k = 3}$, where the plot exhibits a significant slope.

5.3.2 Collective Failure Mitigation

As discussed in section 5.2.2, since models from the same categories have very similar failures, we have considered the possibility of using a single set of generated data, called ***collective_debug_train***, to debug all models within the same categories. To achieve this, we have devised two different settings: 1) we get the failure modes of all models in the same category (e.g. ResNets), and we select k samples from all their failures. Therefore, background failures that occurred more have a higher probability of being chosen for the

Models		Accuracies						
Model category	model name	Accuracy before debugging	**Accuracy of Collective Debugging-type 1** (ours)			**Accuracy of Collective Debugging-type 2** (ours)		
		Test	Test	seed	**heldout**	Test	seed	**heldout**
ResNet	resnet18	0.6236	0.6364	0.2291	**0.2078**	0.6413	0.2636	**0.2128**
	resnet26	0.6593	0.6655	0.2396	**0.2192**	0.6669	0.2185	**0.1853**
	resnet34	0.7017	0.7149	0.2312	**0.2188**	0.7135	0.2391	**0.2178**
	resnet50	0.7631	0.7644	0.2419	**0.2217**	0.7641	0.2325	**0.2105**
	resnet101	0.796	0.8001	0.2474	**0.2226**	0.7999	0.2244	**0.2045**
	resnet152	0.816	0.8182	0.2509	**0.2317**	0.8188	0.2253	**0.2081**
DINO	ViTs8	0.6977	0.70001	0.2729	**0.2488**	0.70008	0.3117	**0.2494**
	ViTs16	0.649	0.6522	0.2673	**0.2394**	0.6504	0.2563	**0.2267**
	ViTb8	0.7101	0.7125	0.2913	**0.2509**	0.7117	0.2903	**0.2540**
	ViTb16	0.6832	0.6840	0.2985	**0.2467**	0.6833	0.2855	**0.2488**

Table 3: Collective_debug_train method's results.

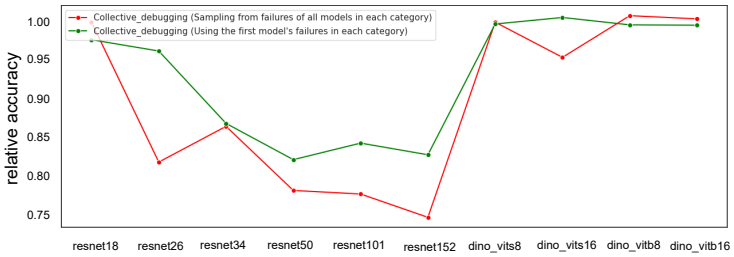


Figure 6: Comparison of resolved failures between collective_debugging and individual_debugging as a percentage. Relative accuracy is the ratio of the combined debugging's accuracy over individual debugging's accuracy on each specific model.

collective_debug_train. We then use this data to debug individual models in this category. 2) We get the failure modes of only one of the models in a category and then use this to debug all models. This approach is more efficient in terms of time and memory, as it requires running only one model per category. The results for this experiment are shown in table 3. Based on our observations in section 5.2.2, having the same debug_train data for debugging (collective_debug_train), improves the accuracies among all models in the same category. This approach offers greater efficiency as it eliminates the need to generate debug_train data for each individual model. Consequently, it saves us both time and memory that would otherwise be required for storing such data.

In an overview, the **collective** debugging approach showcases the capability to resolve above **75%** of failures corrected by individual debugging (Debugging each model based on its failures) 6.

6 Conclusion

In our project, we've developed a technique to identify failure modes by focusing on a specific category of spurious correlations. We then leverage these detected failures to generate additional samples, allowing the model to learn from and address its shortcomings. We've illustrated the resemblance of failures within a particular model category, highlighting that models with the same architecture share more similar failures. Exploiting this insight, we've devised a method to alleviate failures across all models in a category using a single set of generated data based on the failures of just one model in that category. Our results indicate that this collective debugging approach can resolve over 75% of failures addressed through individual debugging efforts. Our framework empowers users to select the spurious correlation to identify and mitigate, facilitating

the simultaneous debugging of a subset of models with a single (small) auxiliary set of additional data, thereby saving both time and resources.

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pp. 233–242. PMLR, 2017.

Hritik Bansal and Aditya Grover. Leaving reality to imagination: Robust classification via generated datasets. *arXiv preprint arXiv:2302.02503*, 2023.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, and Piotr Bojanowski. Unsupervised learning of visual features by contrasting cluster assignments. In *European Conference on Computer Vision*, pp. 116–132. Springer, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Xiaozhi Chen, Kaustav Kundu, Zhiqiang Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pp. 2146–2156, 2018.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2021.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2019.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021b.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *International Conference on Machine Learning*, 2021.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. *arXiv preprint arXiv:2206.14754*, 2022.

Lu Jiang, Dong Meng, Ioannis Mitliagkas, and J. Zico Kolter. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Advances in Neural Information Processing Systems*, pp. 10154–10163, 2018.

Priyatham Kattakinda and Soheil Feizi. Focus: Familiar objects in common and uncommon settings. *arXiv preprint arXiv:2110.03804*, 2021.

Priyatham Kattakinda, Alexander Levine, and Soheil Feizi. Invariant learning via diffusion dreamed distribution shifts. *arXiv preprint arXiv:2211.10370*, 2022.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pp. 13–23, 2019.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Nilaksh Mithun, Soubhik Biswas, and CV Jawahar. Neural modular network for visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13306–13315, 2020.

Besmira Nushi, Ece Kamar, and Eric Horvitz. Towards accountable ai: Hybrid human-machine analyses for characterizing system failure. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 6, pp. 126–135, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. *arXiv preprint arXiv:2205.09739*, 2022.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *International Conference on Learning Representations (ICLR)*, 2015.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. *Advances in Neural Information Processing Systems*, 34:23359–23373, 2021.

Sahil Singla and Soheil Feizi. Salient imagenet: How to discover spurious features in deep learning? *arXiv preprint arXiv:2110.04301*, 2021.

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

Sahil Singla, Atoosa Malemir Chegini, Mazda Moayeri, and Soheil Feiz. Data-centric debugging: mitigating model failures via targeted data collection. *arXiv preprint arXiv:2211.09859*, 2022.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, and Rob Fergus. Training convolutional networks with noisy labels. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 468–476, 2014.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Olivia Wiles, Isabela Albuquerque, and Sven Gowal. Discovering bugs in vision models using off-the-shelf image generation and captioning. *arXiv preprint arXiv:2208.08831*, 2022.

Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning*, pp. 11205–11216. PMLR, 2021.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 747–763, 2019.

Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv preprint arXiv:2006.09994*, 2020.

Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73:213–238, 2007.

Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics*, 25(1):364–373, 2018a.

Min-Ling Zhang, Xingquan Wu, and Zhi-Hua Zhou. Multi-label learning with missing labels: A probabilistic perspective. *IEEE Transactions on Knowledge and Data Engineering*, 30(3):504–517, 2018b.

Bowen Zhou, Yanhua Wu, Bingbing Ni, Xueying Zhou, Jing Wen, and Jie Zou. Towards mitigating bias in deep learning-based automated skin lesion classification with balanced groups and label correction. *IEEE Journal of Biomedical and Health Informatics*, 24(10):2803–2813, 2020.