White Paper Assistance: A Step Forward Beyond the Shortcut Learning

Abstract

The promising performances of CNNs often overshadow the need to examine whether they are doing in the way we are actually interested. We show through experiments that even over-parameterized models would still solve a dataset by recklessly leveraging spurious correlations, or so-called “shortcuts”. To combat with this unintended propensity, we borrow the idea of printer test page and propose a novel approach called White Paper Assistance. Our proposed method involves the white paper to detect the extent to which the model has preference for certain characterized patterns and alleviates it by forcing the model to make a random guess on the white paper. We show the consistent accuracy improvements that are manifest in various architectures, datasets and combinations with other techniques. Experiments have also demonstrated the versatility of our approach on fine-grained recognition, imbalanced classification and robustness to corruptions.

1 Introduction

Computer Vision is defined as a field of study that seeks to deal with how computers can “see” and gain the understanding from digital images or videos. With the emergence of deep learning, numerous astonishing stories about tremendous performances of CNNs have rapidly spread all over the field, ranging from image classification \cite{19, 41, 49}, to object detection \cite{33, 39}, to semantic segmentation \cite{7, 34}, and to video analysis \cite{35, 43}. However, despite the ever-increasing pace, CNNs share the same vulnerability as the human cognitive system, bias. Supported by numerous studies \cite{1, 13, 14, 30}, models may learn “spurious cues/shortcuts” found in training data, resulting in high accuracy on standard benchmarks but severer failure on more challenging testing conditions.

Intrigued, we decided to perform a toy experiment: to let trained models identify an ice-cream image that they never saw. For human beings, when people have acquired a considerable understanding of one kind of object, they usually would not misidentify another new kind of object with objects they have learned. Extending to this case, since a trained model never learns anything concerned ice-creams, we intend the model to assign same probabilities to all the classes. Unfortunately, the inference results completely derail our expectation. Different models mistake this ice-cream image for different classes with different confidence. We argue that this should be seen as a symptom of shortcut learning.

* University of Electronic Science and Technology of China
† Corresponding author
To counteract this, we draw inspiration from the printer test page and derive an interesting and effective regularizer called White Paper Assistance. In the physical world, when we have doubts about whether our printers exist color cast problems, the first check-up that would help is to print a white paper. Similarly, we test the model’s propensity for unintended patterns by evaluating its performance on a sheet of white paper, and force the model to make a random guess on the white paper to alleviate this propensity. Trained in this way, it turns out that the White Paper Assistance can effectively improve the model’s generalization ability and help produce better performance.

In our experiments, we systematically evaluate the effect and performance of White Paper Assistance. We start by elaborating the behavior of our approach in terms of the evolution of training/testing accuracy and the changes in parameters. Results collectively suggest that White Paper Assistance does alleviate the unintended propensity, and it does not devastate the model in the way it seems to act. Then we experiment with various architectures, different benchmark datasets, different combinations of techniques to show the wide applicability and compatibility of our method. Benefiting from the suppression in unintended dominant patterns, our approach also achieves consistent and significant boosts on performance in fine-grained tasks, imbalanced classifications and robustness against corruptions, highlighting the versatility.

2 Related Work

It has been demonstrated that models may learn spurious shortcut correlations, which may be sufficient to solve a training task but are clearly lack of generalization utility. For example, a model would identify cows in “common” (e.g. pastures) contexts correctly but fail to classify cows in “uncommon” (e.g. beach) contexts [14]. Standard ImageNet-trained models prefer to label a cat image with elephant skin texture as elephant instead of cat [14]. Such phenomenons [37, 47, 40] highly exemplify the contradiction between the shortcut correlations and the human-intended generalization.

Recently studies that relates to shortcut learning mainly focus on two contradictory biases of models, texture bias [12, 5] and shape bias [28, 2]. As pointed by [14], features learned by CNNs would bias toward either shape or texture, depending on the training dataset. They demonstrate that a shape-based representation may be more preferable than a texture-based one. [51] further suggests that shape-texture debiased neural network training, that provides supervisions from both shape and texture, leads to better feature representations. However, these studies both entail extensive efforts when generating cue conflict images. Furthermore, too much attention on the shape and texture bias may overshadow the fact that models would also rely on other feature patterns (e.g. color [22]).

Hence it sounds more reasonable that we should understand the shortcut learning with a more holistic view. In this paper, we argue that the shortcut learning can be viewed as preferences for certain features. Namely, models would overly leverage some features to solve the task, which make these features increasingly dominant. In order to alleviate the excessive reliance of these dominant features, we should at least locate them. How can we detect them? What changes need to be made? We will devote the next two section to present the whole deduction process.

3 A Closer Look at Shortcut Learning

We don’t see things as they are; we see them as we are.

–An Old Proverb

These words give us insight into the predictable irrationalities of the human mind. Individuals always create their own “subjective reality” from their perception. Psychological researches [18, 52, 42, 25] term this systematic, irrational, unconscious error that can dramatically alter the way we perceive the world as “cognitive biases”. Similarly to the behavior of human, convolutional neural networks may also develop their own biases during training, by learning “spurious cues [21]” / “shortcuts [13]” which perform well on the existing test data but would fail dramatically under more general settings.

To deepen and broaden the understanding for these “model biases”, we conducted experiments to evaluate the performance of models in a special scenario. An image of ice-cream was first fed into four ResNet-29 models which have already finished training on CIFAR-10. As we know, ice-cream

3This image is collected from ImageNet, then cropped and resized to fit models
does not belong to any class of CIFAR-10. An ideal model with enough generalization ability should thereby provide an inference with uniform output distribution on each class it learned, to demonstrate it does not mistake this out-of-distribution image as any in-distribution class. However, the results are still far from satisfactory.

As shown in left half of Figure 1 different inference results with different confidence are given by these trained ResNet-29. These experiments provide behavioral evidence in favor of shortcut learning of CNNs. The mismatch between the human-intended and actual inference results suggests that CNNs are prone to recklessly capture abundant shortcut correlations from training data, making it challenging to learn generalized features. By way of illustration, ResNet29_0 is clearly more inclined to classify green images as frog while ResNet29_3 has great confidence that this yellow ice-cream cone is some sort of cat. This analogous cognition may be useful (in the sense that knowing it reduces the validation or testing error) when the amount of data is insufficient, but is clearly lack of generalization ability on large and sufficient data (where rare or atypical instances for each class will make up a significant fraction).

Another intriguing finding is that only three classes arise from the inference results of ResNet-29. We further supplemented the experiments by introducing more models of different kinds, such as ResNet-56, Vgg19_bn, WRN-28-10, and PyramidNet-110-270. Surprisingly, in analogy to last experiment we again have the same concentrated pattern regardless that the performances of models have been improved as depicted in right half of Figure 1. More detailed results are presented in Appendix.A. The fact that such a strong model would still trick us by solving classification tasks in an unintended way, are quite revealing that the tendency to learn “spurious shortcuts” are more decisive of data distribution, rather than the performance or architectures of models.

4 Our Method

Motivation: The Color Cast Problem of the Printer  Having shown that learning shortcuts is a common problem that precludes models from better and intended generalization, let us now focus on how to avoid or at least alleviate it. Recent studies have modeled the data distribution of each class as a mixture of various patterns. Naturally, the shortcut learning can be attributed to the propensity for learning dominant patterns that lack generalization information. Hence it seems that the most reasonable and direct way is to identify which patterns contain shortcuts (like green to frogs) and which patterns should be enhanced (like shapes to animals)? Unfortunately, most patterns that CNNs rely on to classify do not appear in a form amenable to discover. And enhancing specific patterns requires specific expert knowledge, let alone extensive manpower and resources.

Luckily, CNNs are not alone with this issue. Very much like the networks submitting to spurious preference, the printer sometimes may use an unintended color to represent the intended color. In the

Figure 1: The results of “Ice-cream Test”: The top-3 inference results and their corresponding confidence distributions provided by several trained models when testing on a ice-cream image. Results from four different ResNet-29 suggest that CNNs are prone to capture shortcut correlations. Further results from other kinds of models suggest the propensity for learning shortcut is caused by the dataset and is independent of the model architecture or depth.
real world, we call it color cast problem. When a colored image is fed into a printer, the printer has to perceive it and then duplicate it using the right color. The color cast problem thereby indicates the wrong propensity of color using. In practical use, when we suspect that our printers are having color cast problems, we usually let the printer print a white paper. Once this white paper is printed into other colors, the color cast is thereby detected and we need to seek corresponding solutions. The white paper here serves as a prefect indicator of the color cast problem. This common sense motivates us to exploit the use of white paper to regularize the model.

Intuitively, the white paper does not belong to any classes the model have learned from whichever benchmark dataset. A trained model should give an inference result that is almost as if it makes a random guess. Consequently, when discovering a difference between the intended and actual outcome, we could know that the model now has some unintended generalization directions, which should be thought of as a consequence of shortcut learning. To further counteract the detected preference, we use the Kullback-Leibler Divergence to force the classification result to match with our intended outcome. So far, we have briefly introduced the core thinking of how to utilize the white paper to detect and alleviate the preference of model. The contents that follow moves on to the detailed design of our scheme.

**Design**

The paradigm we propose is called White Paper Assistance. It is a conceptually simple and plug-and-play method that can be easily integrated into various CNN models without changing the learning strategy.

The pseudo-code of the White Paper Assistance is shown in Algorithm 1. It runs as follows. In training, we conduct it with a certain probability. For certain epoch from training iteration, the probability to conduct White Paper Assistance is $P$, and $1 - P$ if otherwise. Once applying it, a batch of white paper will be fed into the model and we can obtain the normalized output distribution (using “softmax”) $p$. For multi-class classification with $N$ classes, the ideal prediction probability distribution for the white paper would be $q = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right]$. To measure the match of these two predictions $p$ and $q$, we adopt the Kullback-Leibler Divergence:

$$L_{wp} = \lambda \cdot D_{KL}(p \parallel q)$$  \hspace{1cm} (1)

where $\lambda$ denotes the strength of the White Paper Assistance. Then we repeat this process for $M$ iterations.

**Algorithm 1** Pseudo-code of the White Paper Assistance

1: for each epoch do
2:   Real Images training using original loss function
3:   Update model parameters
4:   initialize $p \leftarrow \text{Rand}(0, 1)$ \hspace{1cm} $\triangleright$ White Paper Assistance starts here.
5:   if $p < P$ then
6:     for each iteration $\in [1, M]$ do
7:       Generate a batch of white picture $W$
8:       $p \leftarrow \text{Model}(W)$ \hspace{1cm} $\triangleright$ White paper training.
9:       Update model parameters by Eq. (1)
10:    end for
11:  end if \hspace{1cm} $\triangleright$ White Paper Assistance ends here.
12: end for

There are two important questions for designing above Algorithm:

Q1. Why choose using the white paper?
Q2. Why repeat the process for multiple iterations?

It is tempting to expect that there would be one or more ideal images that not only do not belong to the distribution of training data, but also are able to detect all the unintended dominant patterns. Alas, to precisely find such images require us knowing which patterns CNNs rely on, which is hard because patterns do not appear in a form amenable to discover . . . so, not a viable option. Intriguingly, over all the alternative option, the solution with the white paper works best. As in Figure 2(a), four candidates were evaluated, namely “Gaussian Noise”, “Ice-cream”, “CIFAR-10”, and “White Paper”. We keep all the other implementation details unchanged and merely modified the images
Figure 2: **Controlled Experiments on Designing White Paper Assistance:** (a) Final test accuracy as a function of different types of images which were used in our scheme to detect and alleviate the model’s propensity to shortcuts. The white paper outperforms the other solutions, indicating the superiority of the white paper brought by its uninformative characteristics. (b) Final test accuracy as a function of iterations that we repeat the process for. The shaded areas represent the minimum and maximum results from 4 runs. Dashed dotted line denotes the baseline accuracy. Our approach receives near optimal performances when the amounts of training on white paper and real images are close to each other (In this case, the mini-batch size of real image training is 391).

while training ResNet-56 on CIFAR-100. Specifically, “Gaussian Noise” experiments represent that we changed the white papers into images sampled from a standard normal distribution. “Ice-cream” denotes the whole ice-cream class of images from ImageNet while “CIFAR-10” denotes that all the images from CIFAR-10 were used for detection. Extensive details to facilitate replication are provided in the Appendix.B.

Even with noise-generated images, there is still a performance boost over the vanilla model. Then with the increasing number of real-world images, the performances get higher. But white paper outperforms all the other solution. A possible explanation for this might be that the uninformative nature of the white paper seems to make it more suitable for detecting spurious dominant patterns, since the lack of semantics itself means no bias towards any pattern. Just like coloring on this white paper, the extent to which some pattern plays a dominant role for a class will be shown on the output distribution of the white paper.

Concerning the second question, we answer it with intuitive reasoning and empirical evidence. Intuitively, a biased model cannot be perfectly amended with ease. Take “polishing” as a supportive example. The removal of unintended oxidization requires polishing on the appearance of an item back and forth, until a smoothing finish is accomplished. To verify the intuition, we conduct a controlled trial that we kept $P = 1$ unchanged while modified $M$ from 50 to 500. Aligned with our expectation, more rounds of schemes lead to a better refinement in learning, which results in a better generalization performance. Usually, we find that the performances often get the highest when $M$ is approximately equal to the times real training images are trained per epoch, namely when the amount of training on the white paper is compatible with the amount of training on the real images.

5 **How does White Paper Assistance work?**

Science aims for understanding. What follows are a series of experiments used to glean insights on the behavior of the White Paper Assistance.

**What does the training with White Paper Assistance look like?** Analysis on the trend of training and testing accuracy is of vital importance to understand the effect of a method. Figure 3 (a) and (b) characterize the change of training and testing accuracy across epochs during training with and without White Paper Assistance. Note that we set $P = 1$, namely White Paper Assistance was conducted after each epoch of real images training. Compared with its counterpart, training with WP (be short for White Paper Assistance) exhibits a slower increasing trend on training accuracy, demonstrating that our approach helps suppress model from overusing shortcuts that could rapidly improve the generalization on training data. Even though the training error can both reach zero regardless of the use of our approach, training with WP achieves a significant performance boost on testing data, demonstrating better generalization ability. Not only that, the use of WP on the later
Figure 3: Behavior of White Paper Assistance: (a, b) The evolution of training/testing accuracy when training ResNet-110 with and without WP. (c) Changes in parameters of real images training and white paper training. We use L1 distance to measure the changes of parameters on the final convolutional layer of ResNet-110 when training WP with $P=1$. (d) Illustration of how we determine which part is more affected by WP. (e) Parameter distributions before and after White Paper Assistance was conducted. This change happened on the final convolutional layer of ResNet-110 at epoch 100. More results of changes or distributions on other layers are present in Appendix.C.

stage of training can still provide further improvement with the model, as evident from the fact that training with WP achieves its best performance after epoch 225.$^5$

It is still worth noting that after each time we conducted multiple iterations of white paper training, the testing accuracy would fall dramatically to around 1%. It is as if the model was guessing wildly at all the testing data. But when we moved on and fed real images, both the training and testing accuracy would restore and continue to rise (as seen from the continuous curves of both training and testing accuracy in Figure 3 (a) and (b)), as if the model was not affected by what just happened. Does the state of model performing random guess is a bad sign? Does this mean that White Paper Assistance is harmful? What happened with the model? Why would the accuracy could be restored? We will devote the next part to analyse the causes of it.

Is white paper training harmful to the model? The ultimate goal of training is to find better parameters of the model. To delve deeper into White Paper Assistance, we turn our attention to parameter changes. First, we need to figure out which part of the model is more affected. A trained ResNet-56 $\mathcal{F}(\theta)$ that has achieved 73.51% accuracy on CIFAR-100 was picked. We use $\mathcal{C}$ and $f$ to denote the parameters of all the convolutional layers and the last fully-connected layer at this moment, respectively. Then we applied White Paper Assistance on $\mathcal{F}(\theta)$ and observed the performance dropping to 1%. We also saved the parameters of the convolutional and fully connected layers of the model at this moment, $\tilde{\mathcal{C}}$ and $\tilde{f}$. To determine which part is more affected, we combined $\mathcal{C}$ with $f$ and combined $\tilde{\mathcal{C}}$ with $\tilde{f}$. As shown in Figure 3 (d), for $\mathcal{F}(\theta)$, if we replaced its fully-connected layer, the accuracy changed little (73.51% → 73.3%), whereas the accuracy would drop dramatically (73.51% → 1%) when we replaced all the convolutional layers’ parameters. These observations suggest that the modifications of WP mainly happen on the convolutional blocks, rather than the last fully-connected-layer.

Then we turn to quantitatively measure the changes of parameters due to White Paper Assistance. Since white paper training and real images training alternately appear when $P=1$, we plot the

---

$^5$In this case, we decay the learning rate by factor 0.1 at epochs 150, 225. The training after epoch 225 often suffers from severe overfitting so that it fails to achieve further improvement.
changes of parameters using L1 (mean absolute error) distance with respect to the epoch. In Figure 3 (c), we observe that changes brought by white paper training are smaller compared to real images training. Figure 3 (e) depicts the distributions of the last convolutional layer’s parameter of ResNet-110 before and after the white paper training at certain epoch. It can be seen that our approach is actually disturbing rather than devastating the distributions of parameters. We believe the fact that the model has not been completely devastated proves that White Paper Assistance does not damage the model, and could explain why the accuracy could be rapidly restored. In addition, these results are strong proof that CNNs are vulnerable – slight perturbations of parameters could bring down the whole generalization ability of the model, or at least it seems that way.

6 Does White Paper Assistance perform better?

The section below shows the experimental results to prove that applying White Paper Assistance results in better performance. All experiments are implemented using Pytorch on 4×GTX1080Ti. If not specified, all experimental results reported are averaged over 4 runs. Being limited to the space, we supplement the implementation details in Appendix.D.

6.1 Classification

**Table 1:** Top-1 error rates(%) over different architectures.

| Model          | vanilla ±Δ | + Ours ±Δ |
|----------------|------------|-----------|
| Resnet-110     | 25.65 ± 0.07| 23.65 ± 0.10 |
| Resnet-164     | 24.26 ± 0.06| 22.89 ± 0.08 |
| SeResNet-110   | 23.36 ± 0.09| 22.37 ± 0.04 |
| WRN-28-10      | 21.18 ± 0.01| 19.80 ± 0.04 |
| DenseNet-100-12| 22.78 ± 0.07| 22.02 ± 0.02 |
| ResNext-29, 8×64| 20.68 ± 0.16| 19.41 ± 0.12 |
| PyramidNet-110-270| 18.62 ± 0.03| 17.96 ± 0.08 |
| PyramidNet-200-240| 16.77 ± 0.07| 16.13 ± 0.12 |

White Paper Assistance performs better across different model architectures. As discussed before, White Paper Assistance enjoys a “plug-and-play” property, namely it can be directly amendable to almost any CNNs without any changes needed on the network architecture. To prove it from an empirical standpoint, several different architectures were used to evaluate the effectiveness of our approach, namely, ResNet [19], SeNet [23], Wide ResNet [50], PyramidNet [17], DenseNet [24], and ResNet [48]. All the experiments were performed on CIFAR-100 [29], one of the most extensively studied classification benchmark tasks. In Table 1 we observe that White Paper Assistance consistently outperforms the baseline scheme. For example, our approach achieves 76.35% accuracy when applying Resnet-110 model, which is a 2% improvement over vanilla training. This wide applicability significantly promotes its value in practice and implies, once again, that learning shortcut cues is a common problem that CNNs cannot easily get rid of without special treatment like White Paper Assistance.

**Table 2:** Top-1 error rates(%) on different benchmark datasets.

| Dataset        | Model      | vanilla ±Δ | + Ours ±Δ |
|----------------|------------|------------|-----------|
| SVHN           | ResNet-110 | 3.57 ± 0.02| 3.28 ± 0.01 |
| CIFAR-10       | ResNet-110 | 5.61 ± 0.07| 4.94 ± 0.04 |
| tiny-ImageNet  | ResNet-110 | 37.66 ± 0.18| 35.58 ± 0.05 |

White Paper Assistance gives better performance on various benchmarks. We next performed experiments with different benchmark datasets. The following three benchmark datasets were used: SVHN [56], CIFAR-10 [29], tiny-ImageNet. As shown in Table 2, White Paper Assistance leads to better error rates in all the cases. Such wide applicability to datasets reveals that it is nearly impossible to create a shortcut-free dataset, which further proves the importance of avoiding reliance on unintended shortcuts.

**White Paper Assistance leads to further improvements over other techniques.** In practical use, it is common to use multiple techniques simultaneously in the hope of a higher boost on performance. Thus it imposes very high demands in term of compatibility for a technique. With this in mind, we then applied White Paper Assistance with several widely used techniques, Mixup [51], AutoAugment [8], FastAutoAugment [32], and Label Smoothing [44] into ResNet-110. As reported in Table 5 our
approach consistently makes further improvements, which is reasonable as the training of white paper is independent of the training process of real images.

Table 3: Top-1 error rates(%) over different techniques.

| Method                              | vanilla   | + Ours    |
|-------------------------------------|-----------|-----------|
| + White Paper Assistance            | 23.65 ± 0.09 | -         |
| + Mixup                             | 22.13 ± 0.10 | 21.80 ± 0.23 |
| + AutoAugment                       | 21.70 ± 0.03 | 21.12 ± 0.29 |
| + FastAutoAugment                   | 22.46 ± 0.05 | 21.73 ± 0.15 |
| + Label Smoothing                   | 24.68 ± 0.07 | 23.41 ± 0.07 |

6.2 Fine-grained Recognition

Fine-grained recognition is a highly challenging task that aims at discriminating between subcategories within a general category. Since collecting fine-grained samples often requires expert-level domain knowledge, most fine-grained datasets are hard to extend to large scale. Namely, in a fine-grained recognition task, the model is required to learn discriminative features, rather than general features of a class from a few images. Naturally, while doing so, it becomes easier for the model to capture shortcut correlations due to the small-scale training data. With this in mind, we test the effectiveness of White Paper Assistance on several fine-grained benchmark datasets, CUB-200-2011 [46], StandfordDogs [26], StandfordCars [27].

Specifically, we trained the model from scratch and executed training for 300 epochs to ensure convergence. Other detailed implementation settings are supplemented in Appendix.D. Table 4 presents the results obtained with and without White Paper Assistance. For all the cases, White Paper Assistance achieves significant boosts over vanilla settings. We want to note again that unlike other works which rely on external data to help facilitate recognition, our approach only introduces white paper, which actually does not contain semantic information, into the training scheme. These improvements highlight the importance of alleviating shortcut cues when training.

Table 4: Top-1 error rates(%) on different fine-grained tasks.

| Method   | CUB vanilla | + Ours | Standford Cars vanilla | + Ours | Standford Dogs vanilla | + Ours |
|----------|-------------|--------|------------------------|--------|------------------------|--------|
| ResNet-18| 40.09 ± 0.39 | 35.64 ± 0.23 | 16.42 ± 0.39 | 12.75 ± 0.67 | 47.97 ± 0.35 | 42.94 ± 0.31 |
| ResNet-50| 32.36 ± 0.18 | 28.65 ± 0.05 | 12.12 ± 0.02 | 9.87 ± 0.12 | 38.79 ± 0.74 | 34.55 ± 0.82 |

6.3 Imbalanced Classification

The benchmark datasets we use above all exhibit roughly uniform distributions of class labels. But it is always prohibitively expensive to construct a real-world dataset with proper balance among classes, which explains the long-tailed label distributions that most real-world large-scale datasets [10, 16] have. On these datasets, a few dominant classes hold a large number of samples while a few other classes only possess relatively few samples. Conceivably, with the severe imbalance among the number of classes comes a severer imbalance among patterns, where White Paper Assistance can help. To verify our conjecture, we created the long-tailed version of CIFAR-10 and CIFAR-100 as [6], then validated the performance with and without our approach. We also include a combination of our approach with two other specialized approaches, CB-Focal [9] and LDAM [6]. We strictly use the same parameter settings of [6].

We report the top-1 validation errors of various methods and combinations for long-tailed CIFAR-100 and CIFAR-10 in Table 5. We observe that White Paper Assistance alone can already improve over the vanilla setting, and the combinations of our approach with CB-Focal and LDAM achieve better performance gains, demonstrating both the versatility on imbalanced classification and the
compatibility with other techniques. Overall, these observations strengthen the idea that White Paper Assistance is of great use to solve or alleviate this kind of pattern imbalance problems.

Table 5: Top-1 error rates(%) of ResNet-32 on long-tailed CIFAR-10 and CIFAR-100. The imbalance ratio denotes the ratio between the numbers of samples of the most and least frequent classes.

| Imbalance Ratio($N_{max}/N_{min}$) | Long-tailed CIFAR10 | Long-tailed CIFAR100 |
|-----------------------------------|---------------------|----------------------|
| CE(Baseline)                      | 28.71 ± 0.71        | 61.24 ± 0.05         |
| CE(Baseline) + WP                 | 27.68 ± 0.40        | 60.15 ± 0.07         |
| CB-Focal                          | 27.86 ± 0.16        | 61.51 ± 0.20         |
| CB-Focal + WP                     | 26.79 ± 0.07        | 60.04 ± 0.04         |
| LDAM-DRW                          | 22.59 ± 0.08        | 57.43 ± 0.21         |
| LDAM-DRW + WP                     | 21.15 ± 0.21        | 55.13 ± 0.24         |

6.4 Calibration

As a thought experiment, consider what are the consequences of the model suffering from shortcut learning. The model which has a preference for shortcut correlations usually would become quite good at capturing patterns related to shortcuts and then give an inference result confidently. Since shortcut patterns themselves represent the gold standard for this model. Once the model detects patterns analogous to shortcuts, it would directly jump to the conclusion without a second check on other patterns that may have more generalization utilities. As we can also see in Table 7, even though all the models are being wrong, they are still highly confident in their judgments so that all the confidence scores for top-1 class are greater than 50%. In other words, the fact that modern neural networks are no longer well-calibrated may be due to the ever-increasing propensity of models for shortcut learning.

Figure 4: Reliability diagram of ResNet-110/CIFAR-100 with and without White Paper Assistance.

Therefore, we compare the distribution of prediction confidence with and without White Paper Assistance. In Figure 4 we observe that the reliability diagram slope of White Paper Assistance is much closer to a slope of 1 than baseline. Moreover, the expected calibration error(ECE) of the model trained with White Paper Assistance is lower. These results collectively demonstrate that the model trained with White Paper Assistance is more well-calibrated. We believe that the improvements on confidence estimates are likely to be related to the mitigation on the reliance of shortcut learning.

6.5 Robustness

It is well known that the human vision system is not easily fooled by small changes in query images, whereas existing deep learning models may exhibit dramatic performance decline. This proves
again that the deep learning vision systems do not actually solve a task in the way we intend them to, hence should be viewed as a symptom of models learning shortcuts. In order to check whether White Paper Assistance alleviates the shortcut learning indeed, we evaluate the robustness to common corruptions of our approach on tiny-ImageNet-C [20]. Specifically, we compare the classifiers’ performance with and without WP across five corruption severity levels on each type of given corruption. Since each model performs differently on its own, each result was averaged by four runs. We have written the detailed calculations in the Appendix.E.

The results are shown in Table 6. As expected, White Paper Assistance clearly improves baseline’s robustness against all the corruptions universally. These consistent improvements imply the potential of our approach to generalize to other untested types of corruptions. It’s worth noting again that these improvements are achieved solely through the help of the white paper, which is uninformative and has none relationship with these noisy data. This suggests that our approach is indeed able to improve the robustness, and therefore implies that White Paper Assistance can have an effect on avoiding the excessive reliance of shortcuts.

Table 6: Corruption errors of tiny-ImageNet-C on different corruptions across five corruption severity levels. All metrics are top-1 error rates (for corrupted test sets, we average for 5-severity levels in four runs on ResNet-110).

| Method | NOISE | BLUR | WEATHER | DIGITAL |
|--------|-------|------|---------|--------|
|        | mCE  |      |         |        |
|        | gaussian | shot | impulse | defocus | glass | motion | zoom | snow | frost | fog | brightness | contrast | elastic | pixelate | jpeg |
| w/o WP | 75.32 | 82.12 | 77.83 | 80.44 | 84.29 | 82.77 | 78.08 | 81.87 | 75.31 | 72.00 | 65.98 | 87.32 | 72.82 | 61.29 | 63.80 |
| w/ WP  | 75.87 | 82.77 | 78.08 | 81.87 | 75.31 | 72.00 | 65.98 | 87.32 | 72.82 | 61.29 | 63.80 |

7 Discussion

Effects of parameters. In Appendix.G, we conducted ablation study in CIFAR-100. Specifically, we first evaluated our approach with different Probability $P$ while setting $\lambda = 1$. Then we inspected the performance with different $\lambda$ while keeping $P = 1$. We draw attention to the fact that White Paper Assistance beats the vanilla settings by a clear margin in all the cases, suggesting that it is robust to parameter changes to a large extent.

Limitation This study was limited by the absence of quantitative evaluation of the extent to which our approach avoids shortcut learning. Many of deep learning’s problems can be seen as different symptoms of shortcut learning. It has been broadly defined and cannot even be testified easily. Hence further studies, visualization techniques and specialized datasets regarding shortcuts are recommended. However, we have to admit that all the signs (improvements on imbalanced classifications, robustness against corruptions, more calibrated estimates, effectiveness on fine-grained recognition and even the wide applicability on architectures/datasets) have qualitatively justified the fact that White Paper Assistance indeed avoids the excessive reliance of models on dominant patterns that lack of generalization utility, which should be viewed as a step forward towards a shortcut-free training, as we claimed in the title.

We hope this study could facilitate the awareness for shortcut learning, and more importantly, open up promising avenues towards fair, robust, deployable and trustworthy deep learning.

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020.
[2] Shi Baifeng, Zhang Dinghuai, Dai Qi, Zhu Zhanxing, Mu Yadong, and Wang Jingdong. Informative dropout for robust representation learning: A shape-bias perspective. In International Conference on Machine Learning, 2020.
[3] Pedro Ballester and Ricardo Araujo. On the performance of googlenet and alexnet applied to sketches. pages 1124–1128, 02 2016.
[4] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. pages 456–473.
[5] W. Brendel and M. Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet.
[6] KaiDi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[8] Ekin Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc Le. Autoaugment: Learning augmentation policies from data, 05 2018.

[9] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. pages 9260–9269, 06 2019.

[10] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. pages 4109–4118, 06 2018.

[11] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc., 2020.

[12] Leon A. Gatys, A. S. Ecker, and M. Bethge. Texture and art with deep neural networks. *Current Opinion in Neurobiology*, 46:178–186, 2017.

[13] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. abs/2004.07780, 2020.

[14] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[15] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Weinberger. On calibration of modern neural networks. 06 2017.

[16] Agrim Gupta, Piotr Dollár, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation, 08 2019.

[17] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[18] Martie Haselton, Daniel Nettle, and Damian Murray. *The Evolution of Cognitive Bias*. 11 2015.

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

[20] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[21] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021.

[22] Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9995–10006. Curran Associates, Inc., 2020.

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 06 2018.
[24] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Weinberger. Densely connected convolutional networks. 07 2017.

[25] Daniel Kahneman and Amos Tversky. On the reality of cognitive illusions. Psychological review, 103:582–91; discussion 592, 08 1996.

[26] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, June 2011.

[27] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia, 2013.

[28] N. Kriegekorte. Deep neural networks: A new framework for modeling biological vision and brain information processing. In Annual Review of Vision Science, volume 1, pages 417–446, 2015.

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. University of Toronto, 05 2012.

[30] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. Nature Communications, 10(1096):1096–1104, 2019.

[31] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan L. Yuille, and Cihang Xie. Shape-texture debiased neural network training.

[32] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer, 2016.

[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015.

[35] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4293–4302, 2016.

[36] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. NIPS, 01 2011.

[37] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 427–436, 12 2014.

[38] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, 2015:2901–2907, 04 2015.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497, 2015.

[40] Marco 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, pages 1135–1144, 08 2016.
[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[42] Glenn Shafer, Daniel Kahneman, Paul Slovic, and Amos Tversky. Judgment under uncertainty: Heuristics and biases. *Journal of the American Statistical Association*, 79:223, 03 1984.

[43] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014.

[44] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and ZB Wojna. Rethinking the inception architecture for computer vision. 06 2016.

[45] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1513–1524. Curran Associates, Inc., 2020.

[46] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[47] Felix Wichmann, Jan Drewes, Pedro Rosas, and Karl Gegenfurtner. Animal detection in natural scenes: Critical features revisited. *Journal of vision*, 10:6.1–27, 04 2010.

[48] Saining Xie, Ross Girshick, Piotr Dollar, Z. Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. pages 5987–5995, 07 2017.

[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019.

[50] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. pages 87.1–87.12, 01 2016.

[51] Hongyi Zhang, Moustapha Cisse, Yann Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. 10 2017.

[52] Yu Zhang, Mark Lewis, Michael Pellon, and Phillip Coleman. A preliminary research on modeling cognitive agents for social environments in multi-agent systems. *AAAI Fall Symposium* - *Technical Report*, pages 116–123, 01 2007.
A Detailed Experiment Results of “Ice-cream” Test

Here we present the detailed inference results of the “Ice-cream” Test in Table 7. As can be seen numerically, nearly all the inference results contain only three classes, frog, cat, and bird. Even there are a few exceptions to this, the confidence scores that models assign to these classes (e.g., truck, horse) are approximately equal to zero. These observations suggest that the propensity of CNNs learning shortcuts stem from the data rather than the model itself, which is reasonable as we humans are also often biases or limited by what we have learned.

Table 7: “Ice-cream” Test: Top-3 inference results and their corresponding confidence scores provided by several trained models when testing on a ice-cream image.

| Model          | Accuracy (%) | Class | Confidence (%) | Class | Confidence (%) | Class | Confidence (%) |
|----------------|--------------|-------|----------------|-------|----------------|-------|----------------|
| ResNet-29_0    | 93.20        | Frog  | 66.26          | Cat   | 30.88          | Bird  | 2.54           |
| ResNet-29_1    | 92.97        | Cat   | 52.48          | Bird  | 30.15          | Bird  | 17.28          |
| ResNet-29_2    | 92.62        | Bird  | 58.68          | Cat   | 39.71          | Frog  | 1.48           |
| ResNet-29_3    | 92.94        | Cat   | 99.75          | Frog  | 0.18           | Bird  | 0.03           |
| ResNet-56_0    | 94.26        | Cat   | 80.44          | Frog  | 14.40          | Bird  | 5.12           |
| ResNet-56_1    | 94.14        | Frog  | 77.99          | Bird  | 17.50          | Cat   | 4.51           |
| ResNet-56_2    | 94.06        | Frog  | 88.22          | Cat   | 7.20           | Bird  | 3.22           |
| ResNet-56_3    | 93.71        | Frog  | 80.36          | Bird  | 18.92          | Cat   | 0.48           |
| Vgg19_bn_0     | 93.19        | Frog  | 80.76          | Bird  | 18.96          | Cat   | 0.21           |
| Vgg19_bn_1     | 92.11        | Frog  | 99.83          | Bird  | 0.12           | Cat   | 0.04           |
| Vgg19_bn_2     | 93.32        | Frog  | 99.91          | Truck | 0.04           | Cat   | 0.02           |
| Vgg19_bn_3     | 93.42        | Frog  | 99.66          | Bird  | 0.30           | Cat   | 0.20           |
| WRN-28-10_0    | 96.40        | Cat   | 57.78          | Bird  | 38.10          | Frog  | 3.64           |
| WRN-28-10_1    | 96.25        | Bird  | 99.92          | Cat   | 0.04           | Frog  | 0.02           |
| WRN-28-10_2    | 96.15        | Bird  | 68.29          | Cat   | 30.52          | Frog  | 0.85           |
| WRN-28-10_3    | 96.12        | Bird  | 49.25          | Cat   | 43.67          | Frog  | 6.06           |
| PyramidNet-110-270_0 | 96.23 | Bird  | 89.37          | Cat   | 10.40          | Frog  | 0.18           |
| PyramidNet-110-270_1 | 96.18 | Frog  | 99.77          | Bird  | 0.20           | Horse | 0.02           |
| PyramidNet-110-270_2 | 96.25 | Bird  | 99.28          | Frog  | 0.48           | Deer  | 0.09           |
| PyramidNet-110-270_3 | 96.33 | Bird  | 99.51          | Frog  | 0.35           | Cat   | 0.10           |

B Explanations on Different Types of Images

Here we provide some further explanations on experiments regarding why choose using the white paper. In these experiments, we kept all the scheme details unchanged but only modified the images fed into the model (as in Algorithm 1, line 7).

Vanilla The baseline settings of training ResNet-56 on CIFAR-100.
Gaussian Noise Images randomly sampled from standard normal distribution.
Ice-cream Ice-cream images randomly sampled from ImageNet(Ice-cream class, n07614500).
CIFAR-10 Images sampled from CIFAR-10.

C More experiments exploring behaviour of White Paper Assistance

To begin with, we extended the experiments in Figure 3 (d) (that determine the part that White Paper Assistance has more effect on) to other models. Two models were used to further demonstrate that White Paper Assistance mainly modifies the convolutional layers and has little effect on the last fully-connected layer of models (Table 8).

In Section 5, we still report that White Paper Assistance would incur smaller changes in parameters and such changes would not devastate the inhere parameter distributions. The observations happen on the last convolutional layer of ResNet-110. Here we present more results about the changes and parameter distributions on other layers in Figure 5.
Table 8: Results on other architectures to suggest that the modifications of White Paper Assistance mainly happens on the convolutional layers.

| Model          | Original | After White Paper Training | Combination #1 | Combination #2 |
|----------------|----------|----------------------------|----------------|----------------|
|                | $C + f$  | $\tilde{C} + \tilde{f}$   | $\tilde{C} + \tilde{f}$ | $\tilde{C} + \tilde{f}$ |
| ResNet-110     | 74.18%   | 1.13%                      | 74.26%         | 1.10%          |
| Pyramid-110-270| 81.31%   | 1.00%                      | 81.30%         | 1.00%          |

Figure 5: **Behavior of White Paper Assistance on other convolutional layer** (a,b) Changes in parameters of real images training and white paper training on other convolutional layers. The shaded areas indicate the standard deviations across 4 independent trails. (c,d) Parameter distributions before and after White Paper Assistance conducted on other convolutional layers of ResNet-110 at epoch 100.

**D Implementation Details**

We describe the training implementation settings in detail.

**D.1 Classification**

**CIFAR-100** For ResNet, SeResNet and ResNext, we set the number of training epochs to 300. The learning rate were set to 0.1 and was decaying by the factor of 0.1 at epoch 150 and 225. We used SGD optimizer, and the minibatch size, momentum, weight decay were set to 128, 0.9, and 0.0001, respectively. When training, we set $P = 1$, $\lambda = 1$.

For Wide ResNet, we changed the training epochs to 200. Then the learning rate was decaying by the factor of 0.2 at epoch 60, 120, 160, respectively. When training, we set $P = 1$, $\lambda = 0.5$.

For DenseNet, we changed the mini-batch size to 64 following the common practice. When training, we set $P = 1$, $\lambda = 1$.
For PyramidNet, the learning rate rose to 0.25 and we set $P = 1, \lambda = 0.5$ when training.

For all the experiments on CIFAR-100, we adopted the standard data augmentation techniques including Horizontal Flipping and Random Cropping.

**CIFAR-10** We adopted the same training strategy as on CIFAR100.

**SVHN** We held all the hyper-parameters unchanged but did not adopt any data augmentation techniques. When training, we set $P = 1, \lambda = 0.1$.

**tiny-ImageNet** We adopted the same training strategy as on CIFAR100 except that we changed the mini-batch size to 256.

**Combinations on CIFAR-100** When combined with Mixup, we set $P = 1, \lambda = 1$. For Autoaugment, we adopted the implementation of publicly available code.\(^6\) When training, we set $P = 1, \lambda = 1.5$. For FastAutoaugment, we adopted the implementation of publicly available code.\(^7\) and set $P = 1, \lambda = 1$. For label smooth, we set the smoothing parameter to 0.1 and set $P = 1, \lambda = 1$.

### D.2 Fine-grained Recognition

On fine-grained recognition tasks, ResNet-18 and ResNet-50 were finetuned. We trained them from scratch and we set the training epochs to 300 to ensure convergence. The learning rate was set to 0.1 and decayed by 0.1 at epoch 150, 225. The mini-batch size, momentum, and weight decay were set to 16, 0.9, and 0.0001, respectively. The following contents move on to discuss the data augmentation techniques we used.

**CUB-200-2011 and Stanford Cars** For these two datasets, we first resized images to $600 \times 600$ and cropped them to $448 \times 448$. Then we adopted the Random Horizontal Flipping.

**Stanford Dogs** For the Stanford Dogs, we resized images to $256 \times 256$ and cropped them to $224 \times 224$. Then we also adopted the Random Horizontal Flipping.

### D.3 Imbalanced Classification

Both Long-tailed CIFAR-100 and CIFAR-10 follow an exponential decay in sample size across different classes. The ratio $\rho$ is used to denote the ratio between sample sizes of the most frequent and least frequent class. Figure 6 depicts the sample distributions of Long-tailed CIFAR-100.

![Figure 6: Number of training examples per class in Long-tailed CIFAR-100.](https://github.com/DeepVoltaire/AutoAugment/blob/master/autoaugment.py)

For implementation details, we adopted the same setting as those suggested in \(^8\)

\(^6\) https://github.com/DeepVoltaire/AutoAugment/blob/master/autoaugment.py
\(^7\) https://github.com/ildoonet/cutmix/blob/master/autoaug/archive.py
\(^8\) https://github.com/kaidic/LDAM-DRW

---

16
E Metrics and Comprehensive Results on Robustness

To evaluate the performance of White Paper Assistance on robustness to common corruptions, we refer to the metrics used in tiny-ImageNet-C. Specifically, when evaluating performances of baselines, we took four trained classifiers $f$ trained with vanilla settings. Then we tested the classifier on each corruption type $c$ at each level of severity $s \in \{1, 2, 3, 4, 5\}$. We used $E^f_{s,c}$ to denote the top-1 error rates. Then the average corruption error of all baseline models in corruption $c$ should be:

$$CE_c = \frac{1}{4} \sum_{f=1}^{4} \sum_{s=1}^{5} E^f_{s,c}$$

Then we aggregated all the $CE_c$ to compute the mean corruption error values to all the corruption:

$$mCE = \frac{1}{15} \sum_{c=1}^{15} CE_c$$

Here, we report the results of all the models tested in Table 9.

Table 9: Comprehensive corruption error results of baseline and White Paper Assistance on single model. "Clean" here denote the top-1 error rate of the model on the clean version of tiny-ImageNet.

| Model       | Clean   | Clean avgCE | gaussian | shot | impulse | defocus | glass | motion | snow | loot | fog | brightness | contrast | elastic | pixelate | jpeg |
|-------------|---------|-------------|----------|------|---------|---------|-------|--------|------|------|-----|------------|----------|---------|----------|-----|
| ResNet-110  | 37.41   | 37.29       | 32.05    | 37.68| 31.07   | 86.28   | 83.91 | 79.92  | 83.91| 77.08| 73.29| 69.23     | 87.70    | 75.32   | 62.31    | 66.88|
| ResNet-110  | 37.21   | 37.41       | 33.16    | 38.90| 31.85   | 85.64   | 83.97 | 80.27  | 83.56| 78.72| 75.51| 76.16     | 88.97    | 75.35   | 64.96    | 67.75|
| ResNet-110  | 37.60   | 37.70       | 31.04    | 37.90| 31.85   | 87.14   | 84.29 | 81.52  | 85.53| 78.53| 74.65| 74.07     | 70.95    | 68.56   | 76.53    | 66.37|
| ResNet-110  | 38.11   | 38.22       | 31.50    | 39.90| 32.10   | 87.67   | 84.90 | 81.60  | 85.60| 78.18| 73.28| 76.38     | 70.14    | 69.60   | 76.05    | 64.11|
| ResNet-110+WP| 35.39   | 35.60       | 31.04    | 37.90| 31.85   | 87.14   | 84.29 | 81.52  | 85.53| 78.53| 74.65| 74.07     | 70.95    | 68.56   | 76.53    | 66.37|
| ResNet-110+WP| 35.60   | 35.76       | 31.04    | 37.90| 31.85   | 87.14   | 84.29 | 81.52  | 85.53| 78.53| 74.65| 74.07     | 70.95    | 68.56   | 76.53    | 66.37|
| ResNet-110+WP| 35.76   | 35.96       | 31.04    | 37.90| 31.85   | 87.14   | 84.29 | 81.52  | 85.53| 78.53| 74.65| 74.07     | 70.95    | 68.56   | 76.53    | 66.37|
| ResNet-110+WP| 35.96   | 36.22       | 31.04    | 37.90| 31.85   | 87.14   | 84.29 | 81.52  | 85.53| 78.53| 74.65| 74.07     | 70.95    | 68.56   | 76.53    | 66.37|

F Effects of Parameters

We evaluated the effect of probability $P$ and strength $\lambda$ when training ResNet-56 in CIFAR-100 using the aforementioned settings. We first inspected the performances for different choices of $P \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ when $\lambda = 1$. Specifically, $P = 0$ denotes the baseline without applying White Paper Assistance. In Figure 7 (a), we observe that White Paper Assistance consistently achieves a performance boost even in a small participation rate. Then we turned our attention to another hyper-parameter $\lambda$ that also plays an important role during training. Here we tried different choices with $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8\}$ while keeping $P = 1$. Figure 7 (b) characterizes the evolution of performances on varying $\lambda$. The best performance can be achieved when $\lambda$ is set to 1.