

# Minimizing Data, Maximizing Performance: Generative Examples for Continual Task Learning

## Supplementary Material

### 0.1. CAPER Explanation

In this section, we first introduce a brief list of notations, followed by a detailed explanation of the CAPER removal method.

Table 1. Summary of Notation

Symbol	Description
$f$	Neural network model (classifier)
$l$	Cross entropy loss function
$\mathcal{L}(\theta)$	Loss function parameterized by the model weights
$f^{(l)}(x_i)$	Feature map of sample $x_i$ at layer $l$
$H(\cdot)$	Projection function for feature maps
$\Delta f^{(i)}$	Feature distance between clean and perturbed inputs
$\Delta \hat{f}^{(i,q)}$	Normalized feature distance for channel $q$ of sample $i$
$O^{(l)}$	Number of channels (output dimensions) at layer $l$
$\epsilon_i$	Instability score for sample $i$ in CAPER
$\gamma$	Percentile threshold in CAPER for selecting unstable samples
$\delta_i$	Gaussian noise perturbation added to sample $x_i$
$\sigma$	Standard deviation of Gaussian noise added in CAPER
$m_i$	Binary mask to indicate whether a sample is kept ( $m_i = 1$ ) or removed ( $m_i = 0$ )

In [1], the authors proposed a sample removal strategy to remove those samples that are susceptible to noise based on the difference in feature maps when a small perturbation is added to the samples.

As outlined in Algorithm 1, for each sample, features from a given layer are extracted for both clean and noise-perturbed inputs. These features are projected into a lower-dimensional space, and the feature distance between clean and noisy inputs is measured. Samples exhibiting higher averaged channel-wise perturbation distance are considered unstable and thus more likely to degrade learning; such samples are removed based on a threshold  $\gamma$ .

#### Algorithm 1 CAPER for Redundant Sample Removal - One Task

**Input:** DNN, Layer  $l$ ,  $\gamma$ , Epoch  $\mathcal{T}$ , Noise perturbation  $\sigma$ ,  $N$  Samples

- 1 Train the initialized network using the complete training dataset for  $\mathcal{T}$  epochs
- 2 Capture features from layer  $l$  of the DNN for original input images  $\{x_i\}_{i=1}^N$  and noise-perturbed images  $\{x_i + \delta_i\}_{i=1}^N$ , where  $\delta_i \sim N(0, \sigma^2)$ .

3 **for each sample**  $i \in \{1, 2, \dots, N\}$  **do**

- 4 Compute features  $f^{(l)}(x_i)$  and  $f^{(l)}(x_i + \delta_i)$  for layer  $l$
- 5 Apply projection function  $H$  to map features into a lower-dimensional space
- 6 Compute feature distance

$$\Delta f^{(i)} = \|H(f^{(l)}(x_i)) - H(f^{(l)}(x_i + \delta_i))\|_2$$

- 7 Normalize distances  $\Delta f^{(i)}$  on a channel-wise basis using:

$$\Delta \hat{f}^{(i,q)} = \frac{\Delta f^{(i,q)} - \min_{n \in \{1, \dots, N\}} \Delta f^{(n,q)}}{\max_{n \in \{1, \dots, N\}} \Delta f^{(n,q)} - \min_{n \in \{1, \dots, N\}} \Delta f^{(n,q)}}$$

where  $\Delta \hat{f}^{(i,q)} \in [0, 1]$ .

- 8 Calculate the instability of a sample using:

$$\epsilon_i = \frac{\sum_{q=1}^{O^{(l)}} \Delta \hat{f}^{(i,q)}}{O^{(l)}}$$

- 9 Use  $\epsilon_i$  of each sample to identify and remove susceptible samples using the following mask:

$$m_i = \begin{cases} 0 & \text{if } \epsilon_i \text{ is in the top } \gamma \text{ values of } \epsilon \\ 1 & \text{O.W} \end{cases}$$

where  $\epsilon = \{\epsilon_1, \dots, \epsilon_N\}$ .

Despite its simplicity and effectiveness, CAPER has limitations. It relies solely on perturbations caused by Gaussian noise, which does not generalize well to more diverse or real-world data corruptions such as blur, saturation, or adversarial attacks like PGD. Additionally, based on our experiments, CAPER does not perform well in continual learning settings, particularly when tasks with significantly different distributions are encountered sequentially. Moreover, by making classes more balanced after removal, our

026 implementation performs better on individual tasks than the  
027 method in [8]. However, it still under-performs in both ac-  
028 curacy and robustness compared to our loss-based sample  
029 removal strategy.

## 030 1. Additional Experiments

031 To further validate our proposed method in terms of both  
032 accuracy and adversarial robustness, we have done more  
033 experiments when images are corrupted by either applying  
034 Gaussian Noise, Gaussian Blur, Saturate, or Rotate. These  
035 experiments are conducted on both datasets of synthetic  
036 data, mentioned in the first three parts, and on the MPC  
037 benchmark using VGG16 in the last part.

### 038 1.1. Single-Generator Dataset Comparison with 039 Sample Removal

040 To build off of Figure 6 in the main paper, we consider  
041 the normal and adversarial training settings for the single-  
042 generator variants of GenImage-Disjoint (GenImage-ADM,  
043 GenImage-BigGAN, etc) when applying EpochLoss to re-  
044 move training data in Figure 1. In some cases the applica-  
045 tion of data removal improves accuracy, but generally does  
046 not affect the ranking of which generators perform better or  
047 worse than the others. ADM still performs better than all  
048 other generators, and in some cases better than the natural  
049 image baseline, particularly in adversarial training.

### 050 1.2. Corruption on Synthetic Samples Without 051 Sample Removal

052 In this part, we consider the impact of replacing natural im-  
053 ages with synthetic samples when training on images with  
054 corruptions applied. For these experiments, no removal is  
055 performed. Figure 2 shows that using synthetic data in the  
056 early tasks of CL can improve both accuracy and robustness  
057 compared to training only on natural images.

### 058 1.3. Corruption on Natural Samples With Sample 059 Removal

060 To outline the robustness of our removal strategy in an ad-  
061 versarial setting, we extend our experiments to the use of  
062 both removal and substitution of training data when training  
063 data is corrupted. We begin by analyzing the effectiveness  
064 of our removal model on a CL setup without any data sub-  
065 stitution, meaning that all training images are from the nat-  
066 ural ImageNet dataset. Figure 3 shows the adversarial ac-  
067 curacy under different corruption types, alongside standard  
068 accuracy (denoted as ACC) when the model is trained on  
069 clean, non-adversarial data. These results demonstrate that  
070 EpochLoss outperforms other removal methods and even  
071 the baseline. For these experiments we limit our investi-  
072 gation to the first 3 tasks.

## 073 1.4. Model Generalization and Robustness on Syn- 074 Syn-Syn

075 In this section, we extend our experiments to both robust-  
076 ness and generalization evaluation by substituting all tasks  
077 with synthetic data. Note that, the experiments are done in  
078 3 task CL setup. As seen in Figure 4, in most cases, the use  
079 of synthetic data can outperform the baseline accuracy and  
080 even other removal metrics.

## 081 1.5. Comparison of Data Removal Methods on MPC 082 Benchmark

083 As mentioned in Section 3, to compare our proposed re-  
084 moval strategy with CAPER, we have conducted some ex-  
085 periments on a subset of CIFAR100, including 9000 im-  
086 ages over 20 classes. These experiments are done using  
087 the VGG16 model with the same setup and hyperparamet-  
088 ers mentioned in Section 3. Figure 5 shows the robustness  
089 our removal framework against corrupted images of CI-  
090 FAR100, across most removal percentages and corruption  
091 types. EpochLoss removal consistently matches or exceeds  
092 the performance of CAPER and random removal methods,  
093 even though random removal shows better performance and  
094 robustness than our method dealing with Rotate corruption  
095 either as training corruption or corrupted test images. In  
096 addition, in some cases, EpochLoss outperforms even the  
097 Baseline (no removal) at moderate removal levels.

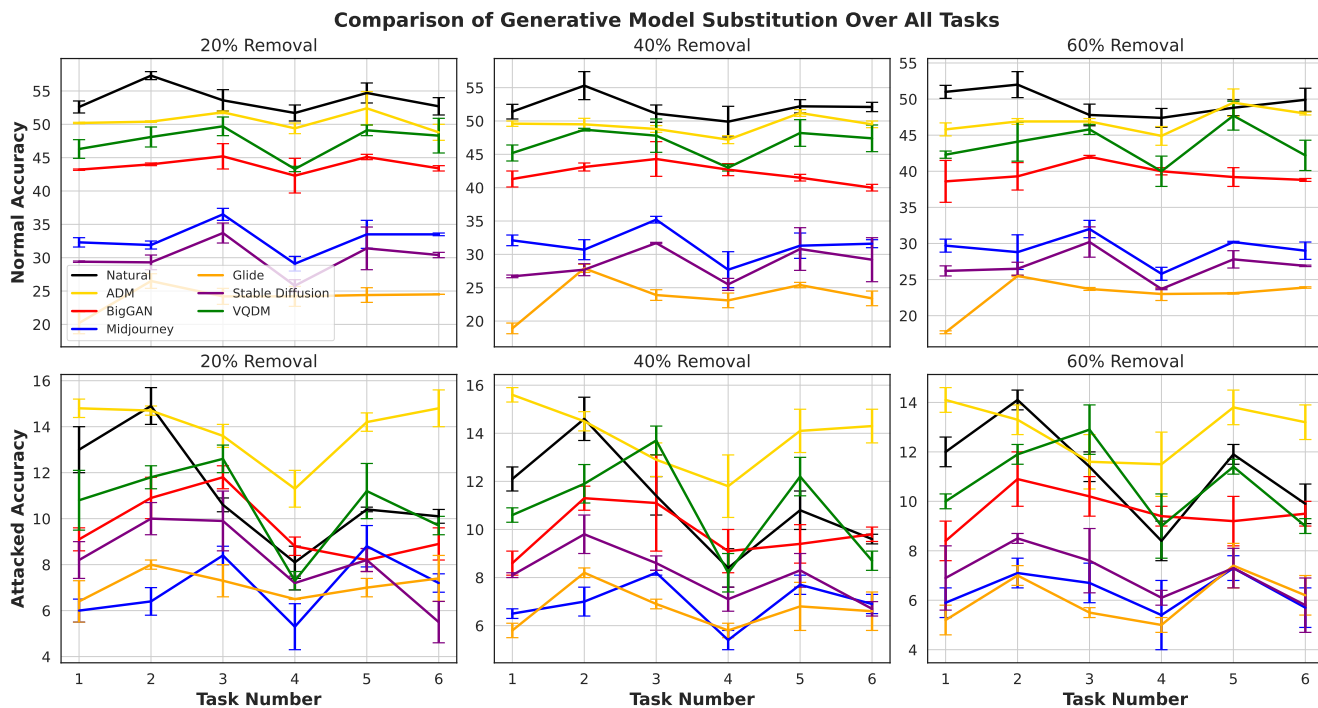


Figure 1. For each single-generator variant of GenImage-Disjoint, we compare the natural test accuracy over all six tasks when removing a portion of training data using EpochLoss. We compare training on synthetic data for all tasks against training on the natural tasks of GenImage-Disjoint. In both the non-adversarial (top) and adversarially trained (bottom) settings, we see that the removal gradually reduces accuracy beyond a certain point, but does not disproportionately affect any one generator.

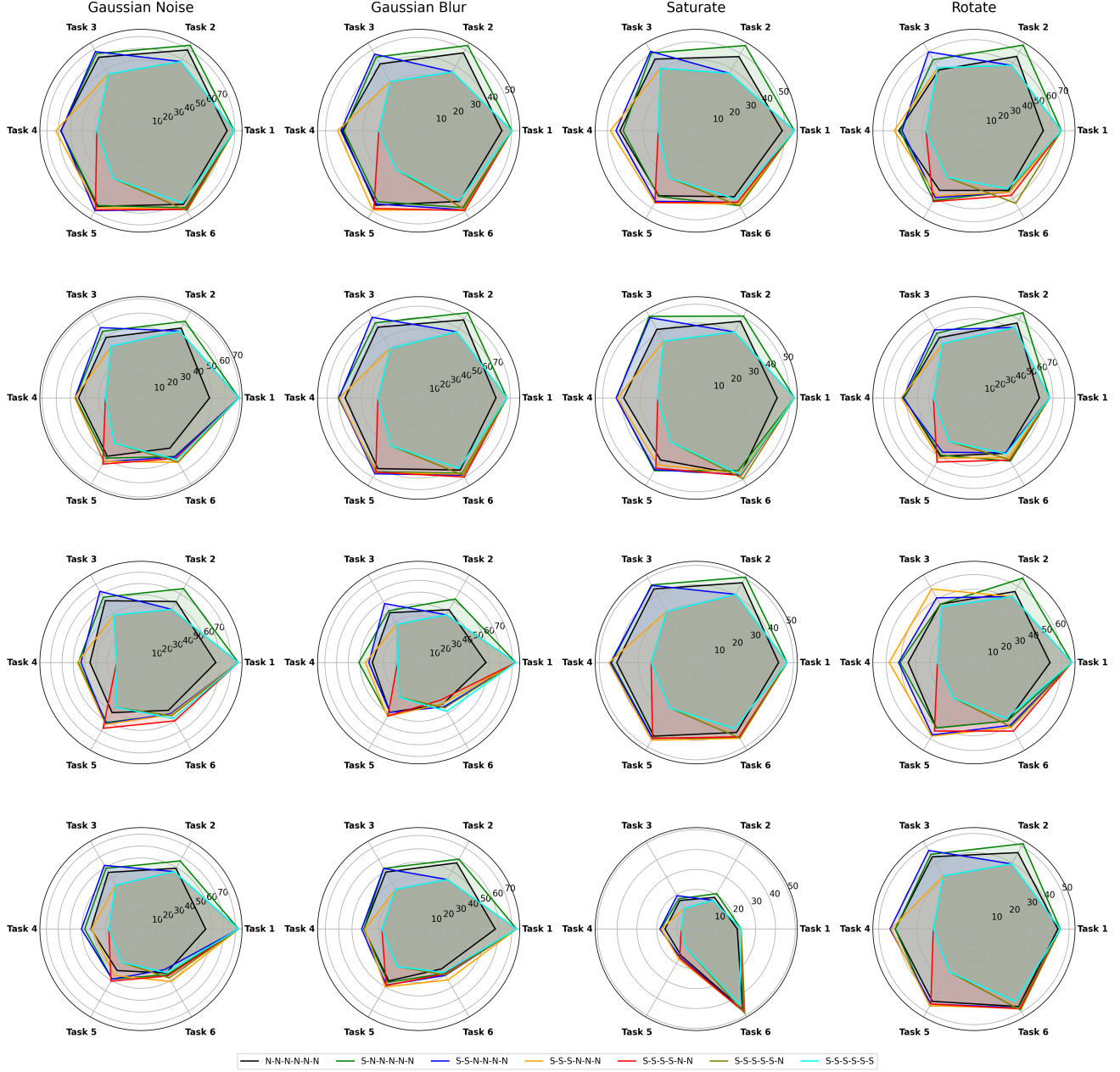


Figure 2. Comparison of adversarial robustness across different task configurations using natural (N) and synthetic images (S), when the model is trained on corrupted images using Gaussian Noise, Gaussian Blur, Saturate, and Rotate, respectively, from top to bottom (rows), whereas the columns correspond to the evaluation on a corruption.

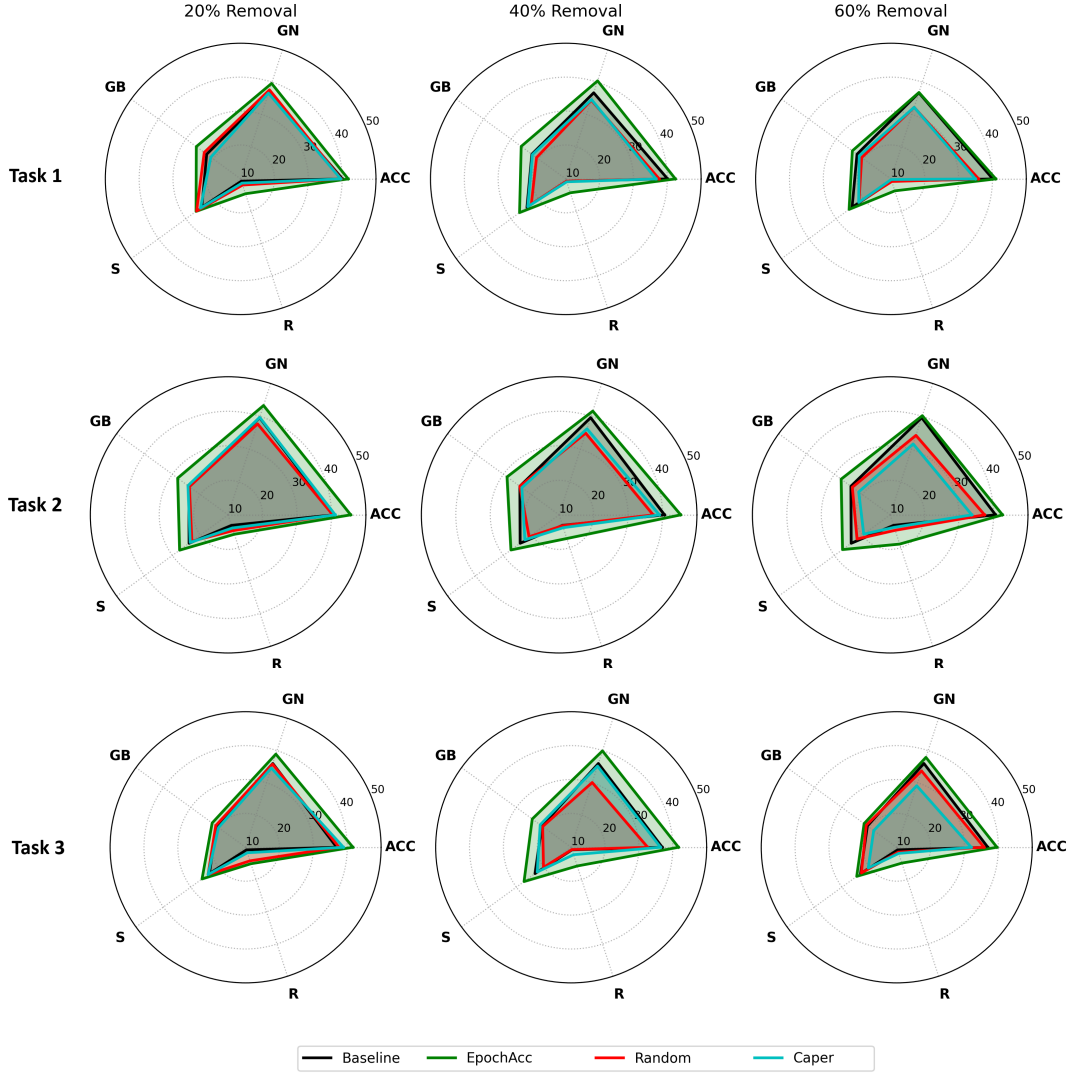


Figure 3. Comparison of standard and adversarial accuracy between EpochLoss and other sample removal methods under various corruption types: Gaussian Noise (GN), Gaussian Blur (GB), Saturation (S), and Rotation (R). The model is trained on Gaussian Noise-corrupted dataset. ACC refers to the standard (non-adversarial) accuracy when trained on clean data. All tasks consist of natural images without synthetic substitution.

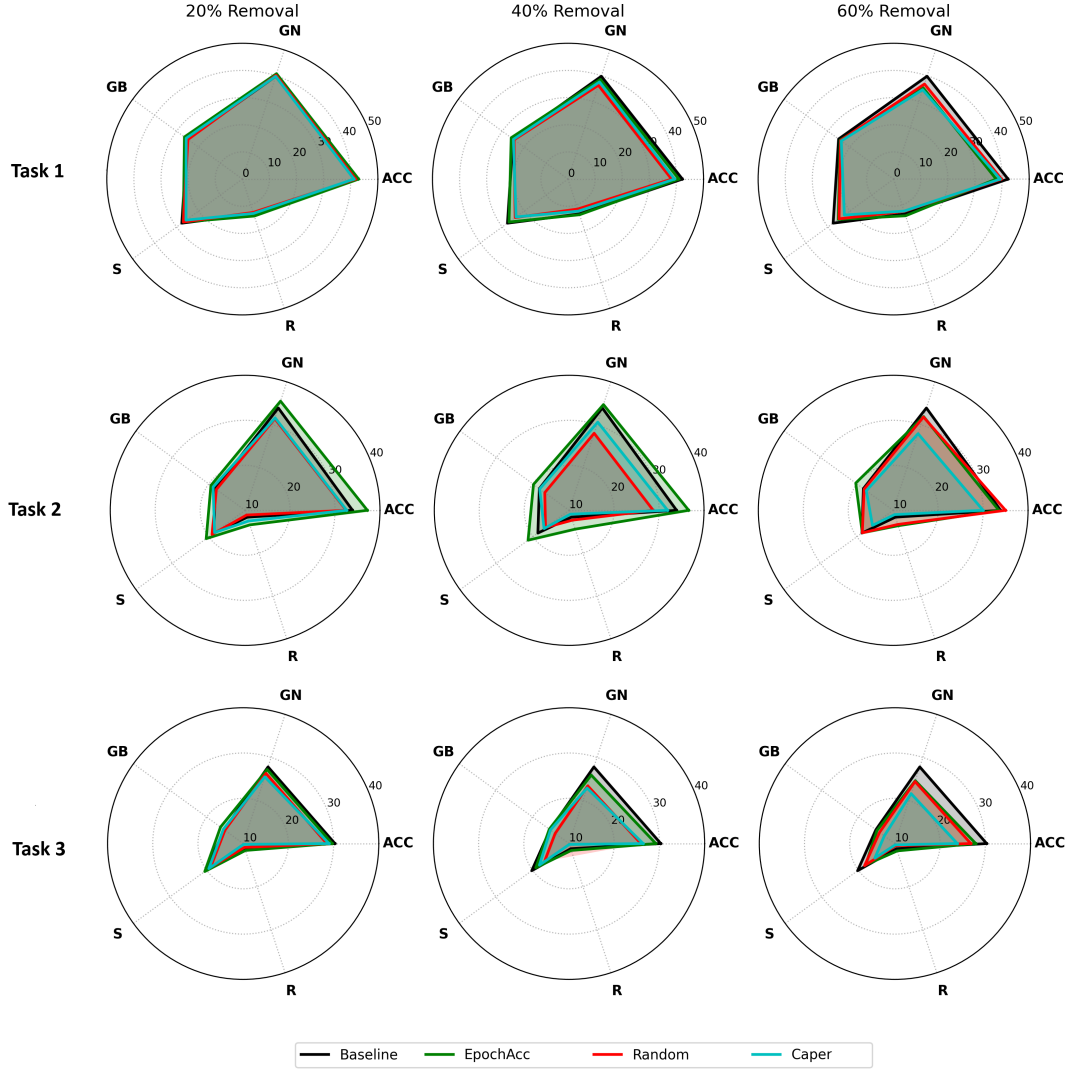


Figure 4. Comparison of standard and adversarial accuracy between EpochLoss and other sample removal methods under various corruption types: Gaussian Noise (GN), Gaussian Blur (GB), Saturation (S), and Rotation (R). The model is trained on Gaussian Noise-corrupted dataset. *ACC* refers to the standard (non-adversarial) accuracy when trained on clean data. All tasks are substituted with synthetic images. From top to bottom: results for Tasks 1 through 3.



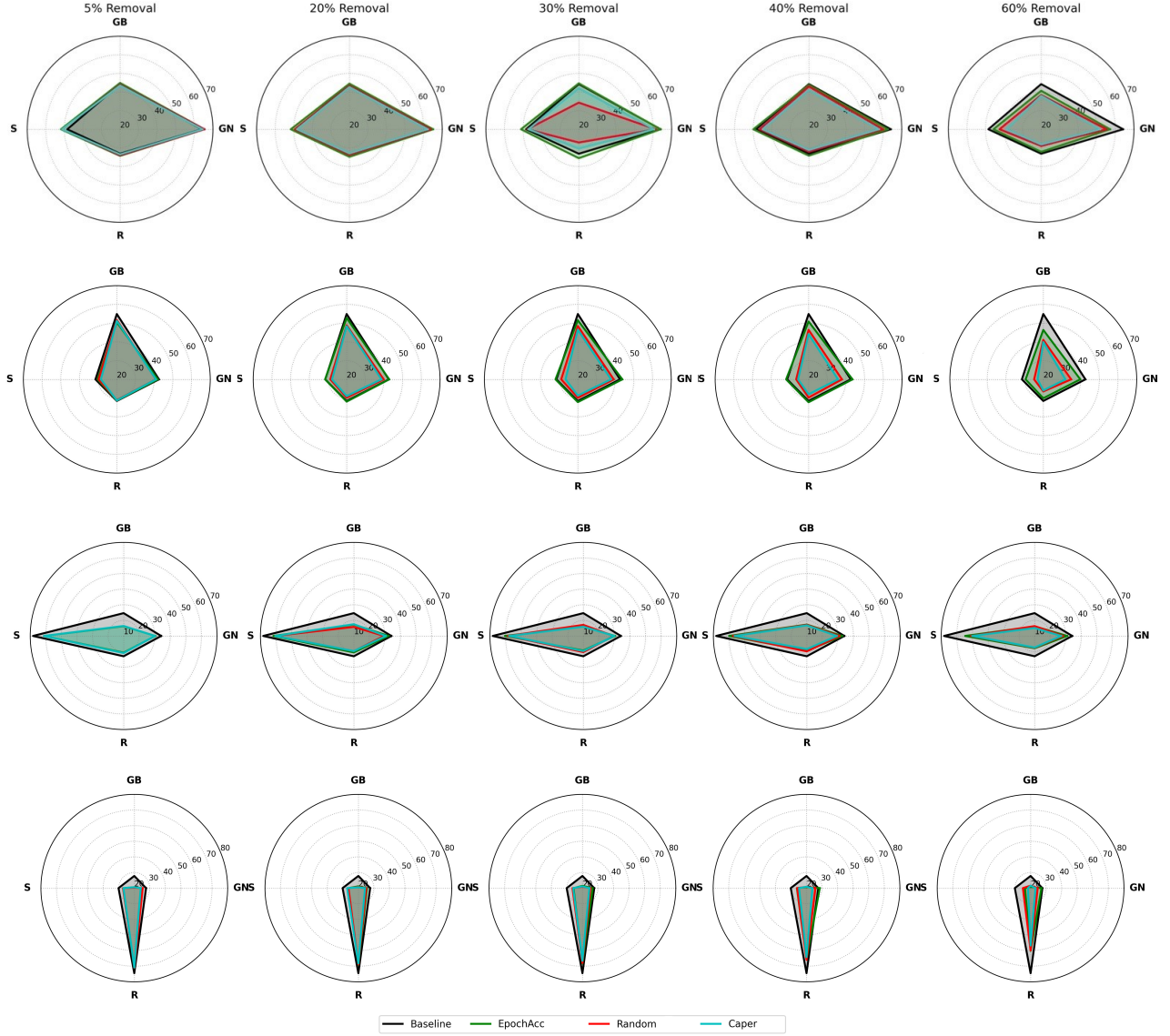


Figure 5. Comparison of adversarial accuracy and robustness between EpochLoss and other removal methods under various corruptions: Gaussian Noise (GN), Gaussian Blur (GB), Saturation (S), and Rotation (R). The VGG16 model is trained on corrupted images of CIFAR100 with 9000 images using each respective corruption, from top to bottom: GN, GB, S, and R.