A APPENDIX

Algorithm 1 Adversarial Training with FOMO

Require: Training data $D = (x_i, y_i)i = 1^n$, number of training epochs N, batch size $\mathcal{B}$, base adversarial training algorithm $\mathcal{A}$ with ϵ, perturbation norm and steps, network f_θ, stable model f_ϕ, warm-up epochs $e_{warm-up}$, number of epochs the network undergoes relearning e_r, decay parameter α_c, s sparsity rate that determines the percentage of parameters to be forgotten in the later layers defined by layer threshold L.

Initialize model parameters θ
for $epoch \leftarrow 1$ to N **do**
  if $epoch > e_{warm-up}$ and $epoch\%e_r == 0$ **then**
    Consolidate(f_θ, f_ϕ, α_c)
    Random_forgetting(f_θ, s) ▷ Randomly reinitialize the parameters in the later layers
  Sample a mini-batch $\mathcal{B} = (x_i, y_i)$ from D
  $\hat{\mathcal{B}}$ = $\mathcal{A}$(f_θ, $\mathcal{L}_{adv}$, ϵ, steps, norm) ▷ adv samples
  $\hat{\mathcal{B}'} \leftarrow f_\theta(\hat{\mathcal{B}})$ ▷ forward pass with adv samples
  $\mathcal{B}' \leftarrow f_\theta(\mathcal{B})$ ▷ forward pass with std samples
  if $epoch > e_{warm-up}$ **then**
    $\hat{\mathcal{B}''} \leftarrow f_\phi(\hat{\mathcal{B}})$ ▷ forward pass using stable model
    $\mathcal{B}'' \leftarrow f_\phi(\mathcal{B})$
    $\mathcal{L}_{CR} = \lambda_1 \cdot D_{KL}(\hat{\mathcal{B}'}||\hat{\mathcal{B}''}) + \lambda_2 \cdot D_{KL}(\mathcal{B}'||\mathcal{B}'')$
    $\mathcal{L}_{FOMO} = \mathcal{L}_{adv} + \mathcal{L}_{CR}$ ▷ refer Eq 2 & Eq 3
  else
    $\mathcal{L}_{FOMO} = \mathcal{L}_{adv}$
  Compute the gradients
  Update the parameters f_θ

A.1 IMPLEMENTATION DETAILS

We follow the standard adversarial training (AT) procedure used in previous research (Wu et al., 2020). The model was trained for a total of 200 epochs using the stochastic gradient descent (SGD) optimization algorithm with a momentum of 0.9, a weight decay of 5×10^{-4}, and an initial learning rate of 0.1. For standard AT, we reduced the learning rate by a factor of 10 at the 100^{th} and 150^{th} epochs, respectively. We applied standard data augmentation techniques, including random cropping with 4-pixel padding and random horizontal flipping, to the CIFAR-10 and CIFAR-100 datasets, while no data augmentation was applied to the SVHN dataset.

For training the other baseline methods (Wu et al., 2020; Chen et al., 2020; Dong et al., 2021), we used the exact same procedure and hyperparameters as specified in those methods. For the FOMO method proposed in Section 3, we began at epoch 105 ($e_{warm-up}$), a little later than the first LR decay where robust overfitting often occurs. For PreActResNet-18, we forgot a fixed s = 3.5% of the parameters in the later layers (Block-3 and Block-4) of the architecture, while for widerResNet-34-10, we forgot 5% as it has a larger capacity to memorize. Each forgetting step was followed by a relearning phase that lasted for $e_r = 5$ epochs. The relationship between s and e_r is studied in Section A.6. For the consolidation step, we chose a decay rate of the stable model of $\dashv_c = 0.999$. During the relearning phase, the stable model through the regularization loss ($\mathcal{L}_{CR}$), and we chose regularization strengths of λ_1 and λ_2 equal to 1. We ran the experiments for three seeds, and the average of the results is reported in the table.

Datasets. For our experiments, we use three datasets: CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011). We randomly split the original training sets for these datasets into a training set and a validation set in a 9:1 ratio. We provide the results for the SVHN dataset in Appendix. Our ablation studies and visualizations are mainly based on the CIFAR-10 dataset.

Baseline. We compare the results against various baseline methods such as vanilla PGD-AT (Madry et al., 2017), TRADES (Zhang et al., 2019), KD+SWA (Chen et al., 2020), PGD-AT+TE (Dong et al., 2021), AWP Wu et al. (2020) that are proposed to mitigate robust overfitting. To assess the model's robust overfitting ability, we compare the robust test accuracy between the best-epoch

Table 5: Comparison of test robustness (%) between MLCAT and FOMO under Autoattack.

Method	CIFAR-10			CIFAR-100		
	Best	Last	Δ	Best	Last	Δ
MLCAT$_{LS}$	28.12	27.03	-1.29	13.41	11.37	-2.04
MLCAT$_{WP}$	50.70	50.32	-0.38	25.86	25.18	-0.68
FOMO	**51.37**	**51.28**	**-0.09**	**27.57**	**27.49**	**-0.08**

Table 6: Test robustness (%) of AT and FOMO across different datasets and threat models.

Threat Model	Method	CIFAR-10			CIFAR-100		
		Best	Last	Δ	Best	Last	Δ
ℓ_∞	PGD-AT	52.32	44.44	-7.88	27.22	20.82	-6.4
	FOMO	**56.68**	**56.46**	**-0.22**	**32.07**	**31.67**	**-0.40**
ℓ_2	PGD-AT	69.15	65.93	-3.22	41.33	35.27	-6.06
	FOMO	**72.69**	**72.28**	**-0.41**	**45.60**	**45.16**	**-0.44**

and the last-epoch. The difference between the best and final robust test accuracy is denoted as Δ. Results are averaged over three seeds. In addition to robust overfitting, achieving an optimal balance between natural and robust test accuracy is crucial for an effective AT method. However, there is currently no standardized method for measuring this trade-off in the adversarial trade-off literature. Therefore, we propose a trade-off measure that offers a formal approach to compare how well different methods maintain this balance. The Trade-off is measured as follows:

$$\text{Trade-off} = \frac{2 \times \mathcal{NA}_L \times \mathcal{RA}_L}{\mathcal{NA}_L + \mathcal{RA}_L} \tag{4}$$

where $\mathcal{NA}_L$ and $\mathcal{RA}_L$ stand for last-epoch natural test accuracy and robust test accuracy, respectively. Implementation details and additional experiments can be found in Appendix.

A.2 Comparison with MLCAT

Yu et al. (2022) proposed a method called MLCAT to alleviate robust overfitting (RO) by analyzing the roles of easy and hard samples and regularizing samples with small loss values using non-robust features. In contrast, we approach this problem from the lens of active forgetting, which provides a different perspective for addressing robust overfitting. Here we compare our work with theirs in detail.

According to the results presented in Table 5, our FOMO approach outperforms MLCAT consistently in terms of robust improvement against AutoAttack across datasets. Additionally, FOMO exhibits lower levels of robust overfitting compared to both variants of MLCAT. MLCAT$_{LS}$ corresponds to loss scaling, and MLCAT$_{WP}$ corresponds to weight perturbation. As a result, FOMO is not only more effective than MLCAT but also shows a reduced tendency toward overfitting. Thus, by iterating periodically between forgetting, relearning, and consolidation during AT brings discernible benefits regarding robustness to AT.

A.3 Extended comparison with baselines on large datasets:

We conducted extensive comparisons with KD-SWA, weight averaging (WA), and model ensembling approaches on CIFAR10/100, SVHN, and Tiny-ImageNet consisting of 64x64 images. FOMO algorithm outperforms these baselines, demonstrating superior robustness and im- proved performance as shown in Table 7. Additionally, as shown in Table 8, FOMO outperforms Subspace adversarial training (Sub-AT) (Li et al., 2022). This shows the effectiveness of FOMO even on larger datasets.

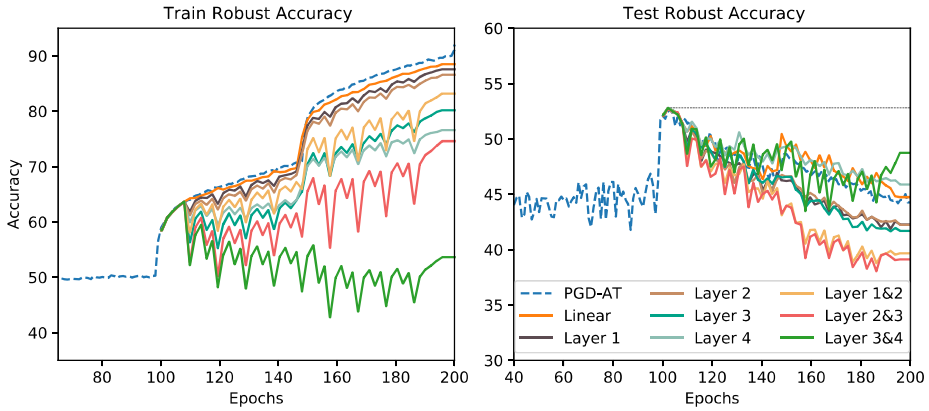
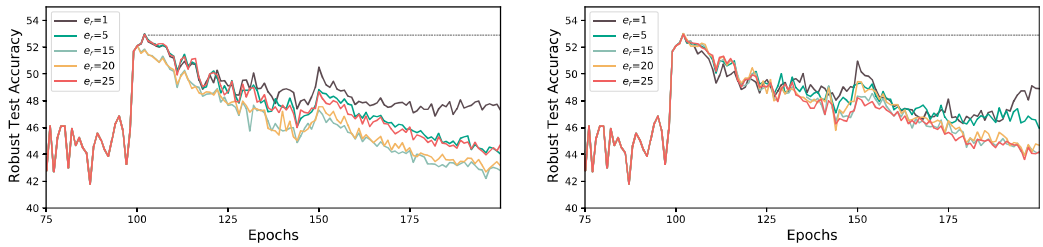


Figure 5: The impact of forgetting 50% of parameters in each layer on robust generalization using PreAct-ResNet-18 on CIFAR-10 is illustrated in the figures. The left figure shows the impact on train robust accuracy and the right figure shows the impact on test robust accuracy. It is evident from the figures that forgetting in the later layers regularizes the train robust accuracy and mitigates robust overfitting when compared to forgetting in the early layers.



(a) Forgetting 3% parameters in later layers.

(b) Forgetting 10% parameters in later layers.

(c) Forgetting 30% parameters in later layers.

(d) Forgetting 50% parameters in later layers.

Figure 6: Study the symbiotic relationship between forgetting and relearning during adversarial training

A.4 Evaluation across Perturbations

We assess the versatility of our proposed FOMO framework by evaluating across perturbation (ℓ_2 norm) using PreActResNet-18 He et al. (2016) on CIFAR-10. Table 6 demonstrates the best and final performance of the FOMO network compared against vanilla adversarial training (PGD-AT) with ℓ_2 perturbation norm. We use ϵ of 128/255 and a step size of 15/255 for evaluation with ℓ_2 perturbation. The training/test attacks are PGD-10/PGD20, respectively. The results demonstrate that FOMO significantly outperforms vanilla adversarial training across perturbations. Therefore, forgetting and relearning are more effective in mitigating robust overfitting across perturbations.

Table 7: Auto-attack on FOMO using the PreActResNet-18.

Method	CIFAR-10		CIFAR-100		Tiny-ImageNet	
	Best	Last	Best	Last	Best	Last
WA	49.92	43.82	25.95	21.02	19.76	15.82
KD+SWA	49.87	49.74	26.04	25.99	19.78	19.76
PGD-AT+TE	50.11	49.14	26.04	25.13	18.16	15.88
FOMO	**51.37**	**51.28**	**27.57**	**27.49**	**20.23**	**19.85**

Table 8: PGD-attack on FOMO against Subspace AT.

Method	CIFAR-10			CIFAR-100		
	Best	Last	Δ	Best	Last	Δ
Sub-AT	52.79	52.31	0.48	27.50	27.02	0.48
FOMO	**56.68**	**56.46**	**-0.22**	**32.07**	**31.67**	**-0.40**

A.5 Effect of forgetting in different layers on Robust overfitting

Training DNNs adversarially often results in a predominant phenomenon known as robust overfitting. Current learning techniques generally analyze learning behavior by treating the network as a whole unit, which disregards the ability of individual layers to learn adversarial data distributions. We suggest that different layers possess unique capacities to learn information, and it is crucial to comprehend these patterns to develop a training scheme that mitigates robust overfitting. Therefore, we investigate the layer-wise characteristics of a network by analyzing the effect of forgetting parameters at different layers on robust forgetting. For this analysis, we use the PreActResNet18 architecture, where each block (4 blocks) is considered a layer along with an additional linear classification layer. Figure 5 demonstrates the effect of forgetting in different layers on the robust train and test accuracy.

Our analysis revealed that forgetting on early layers often results in decreased performance on robust test accuracy. Additionally, it fails to regularize the training accuracy, leading to a larger robust generalization gap. This is possibly due to the lower capacity of earlier layers to accommodate new information. Moreover, since earlier layers learn generalized features compared to later layers, forgetting them results in a loss of generalization. On the other hand, later layers have more capacity to memorize and tend to overfit the training data. Therefore, forgetting layers 3 and 4 leads to a reduced robust generalization gap, which mitigates robust overfitting. By limiting the forgetting process to the last two layers, we aim to regularize the weights in the later layers while retaining more generalized features learned in the earlier layers to facilitate robust generalization.

A.6 Study the symbiotic relationship between forgetting and relearning

Recent advances in cognitive neuroscience have shed light on the interdependent nature of learning and forgetting, with mounting evidence indicating a symbiotic relationship between the two phenomena Bjork & Allen (1970); Bjork & Bjork (2019). In the context of DNNs, the dynamics of forgetting and relearning are of particular interest, as they have been shown to play a critical role in mitigating the deleterious effects of overfitting. Specifically, the percentage of forgotten parameters and the duration of the relearning phase (e_r) are important factors to consider to achieve robust overfitting.

To investigate the impact of forgetting and relearning on overfitting, we varied the percentage of parameters forgotten (3%, 10%, 30%, and 50%) and adjusted the duration of the relearning phase over six different intervals (1, 5, 15, 20, and 25 epochs) within a total training duration of 200 epochs. Figure 6 presents the relationship between forgetting different parameters and the relearning phase during the training process.

Our results suggest that forgetting only a small percentage of parameters can effectively mitigate overfitting when combined with a relatively short relearning phase. However, forgetting many pa-

rameters with short relearning phases can make training DNNs challenging. This implies that the percentage of parameters forgotten directly correlates with the duration of the relearning phase; the higher the percentage of parameters forgotten, the longer the relearning interval required to relearn the necessary information from the previous step. It is, therefore, essential to balance the amount of forgetting and the duration of the relearning phase to enable effective relearning and retain the critical information necessary for optimal performance.