Two is Better than One: Efficient Ensemble Defense for Robust and Compact Models

Supplementary Material

A. Implementation Details

A.1. Additional Experimental Setups

We used VGG16 and ResNet18 architectures for experiments on small-scale datasets, CIFAR-10 and SVHN. Also, we employed ResNet50 as the classifier for additional large-scale datasets, ImageNet and CIFAR-100. The hyperparameter φ was set to 0.01 for small-scale datasets and 0.1 for large-scale datasets. As adversarial training methods, we adopted Madry-AT [28], TRADES-AT [52], and MART-AT [42]. For TRADES-AT and MART-AT, the original regularization values of $\lambda = 6.0$ and $\lambda = 5.0$, respectively, were set. During training, we allocated 100 epochs for AT (pre-training), 20 epochs for pruning, 40 epochs for sub-model fine-tuning, and 80 epochs for ensemble learning.

A.2. Detailed Descriptions on Adversarial Pruning

As described in Sec. 3.1, following HARP [53], we dynamically learned the optimal layer-wise compression rates along with importance scores to construct a pruning mask M that maximizes adversarial robustness while satisfying the target sparsity constraint. The core objective is formulated as a min-max optimization problem, as shown in Eq. (6). Here, $\mathcal{L}_r$ represents the adversarial robustness loss based on adversarial training (AT), and $\mathcal{L}_p$ is the global compression control loss that ensures the overall sparsity aligns with the target compression rate.

To explain the pruning process in greater detail, each layer l is associated with a binary pruning mask $M^{(l)}$, which determines the pruned parameters. This mask is computed using a learnable importance score matrix $I^{(l)}$ and the layer-wise compression rate $c^{(l)}$. A pruning threshold $a^{(l)} = 1 - c^{(l)}$ is derived, and parameters with importance scores below this threshold are pruned:

$$M^{(l)} := \mathbf{1}\left[I^{(l)} > P(a^{(l)}, I^{(l)})\right] \tag{23}$$

Here, $P(a^{(l)}, I^{(l)})$ represents the percentile calculated based on the pruning threshold $a^{(l)}$ and the importance scores $I^{(l)}$. The compression rate of each layer $c^{(l)}$ is parameterized using a learnable compression quota $r^{(l)}$ and a sigmoid-based activation function $g(r)$. The resulting values are constrained within the range $[a_{\text{min}}, 1]$, as expressed below:

$$a^{(l)} = g(r^{(l)}) = (1 - a_{\text{min}}) \cdot \text{sig}(r^{(l)}) + a_{\text{min}} \tag{24}$$

Such dynamic control prevents excessive pruning of any specific layer by ensuring that the pruning threshold satisfies $a^{(l)} \geq a_{\text{min}}$.

To align the overall compression ratio with the target sparsity a_t, the compression loss $\mathcal{L}_p$ adjusts the deviation as follows:

$$L_p(\theta \odot M, a_t) := \max\left(\frac{\|\theta_{\neq 0}\|_1}{a_t \cdot \|\theta\|_1} - 1, 0\right) \tag{25}$$

where $\|\theta_{\neq 0}\|_1$ represents the number of non-zero parameters remaining after pruning.

A.3. Detailed Descriptions on Dynamic Inference Ensemble

As mentioned in Sec. 4.4, the core of Dynamic Inference Ensemble (DIE) is to determine the optimal stopping point z_x for a given sample x, maximizing robustness while ensuring computational efficiency. In more detail, the probability that all models remain active up to and including step t is defined as:

$$S(t) = 1 - \sum_{i=1}^{t-1} q_i \tag{26}$$

which represents the likelihood that all models have participated in inference until step t. The conditional probability q_t of stopping inference at step t, given that inference has not stopped prior, is expressed as:

$$q_t = \frac{S(t) - S(t+1)}{S(t)} \tag{27}$$

The sequential decision-making process at each step aims to optimize the probability z_t to identify z_x, where z_t is derived as:

$$z_t = q_t \prod_{i=1}^{t-1}(1 - q_i) \tag{28}$$

In this context, q_t is computed based on the predictive uncertainty and prediction confidence at each step. The predictive uncertainty is evaluated using the Kullback-Leibler (KL) divergence between the intermediate ensemble predictions at step t and step $t-1$. Specifically, the KL divergence quantifies the change in the probability distributions of the predictions, reflecting how the ensemble output evolves as additional models contribute. Mathematically, the predictive uncertainty at step t can be defined as:

$$Uncertainty_t = KL(\hat{y}_{ens}^{t} || \hat{y}_{ens}^{t-1}) \tag{29}$$

Table 4. Comparison of various AP methods and EED on CIFAR-100 and ImageNet datasets when sparsity rate $s_r = 90\%$

	Setting	CIFAR-100 Clean	PGD	AA	C&W	DF	Speed up	Imagenet Clean	PGD	AA	C&W	DF	Speed up	Prams
Resnet 50	AT[28]	64.37	36.29	28.07	30.18	39.80	1.00x	60.25	32.38	28.79	30.67	34.54	1.00x	25.6M
	R-ADMM[49]	61.38	31.23	21.85	24.04	30.42	2.43x	35.26	14.35	11.01	12.35	13.53	2.41x	2.6M
	HYDRA[35]	62.10	33.52	24.12	26.20	33.94	2.80x	49.44	23.75	19.88	21.60	23.14	2.73x	
	RST[11]	61.14	29.81	20.15	21.38	28.45	**2.98x**	27.09	12.23	10.09	11.22	12.34	**2.96x**	
	MAD[22]	56.88	30.59	21.53	23.55	30.97	2.55x	34.62	14.67	11.24	12.42	13.47	2.60x	
	Flying Bird[2]	60.03	33.18	24.91	24.19	32.26	2.75x	48.39	23.05	18.14	19.33	17.65	2.56x	
	HARP[53]	62.51	33.40	25.36	27.25	34.20	2.87x	55.21	27.10	22.57	24.62	25.57	2.88x	
	TwinRep[24]	62.31	34.08	24.44	**28.92**	33.73	2.86x	52.46	26.73	24.75	24.37	24.68	2.67x	
	EED(ours)	**63.60**	**36.29**	**26.79**	28.01	**37.14**	2.90x	**58.41**	**30.54**	**26.89**	**27.15**	**29.92**	2.84x	
Resnet 101	AT[28]	-	-	-	-	-	-	83.29	62.23	51.34	58.65	47.29	1.00x	44.6M
	Flying Bird[2]	-	-	-	-	-	-	72.56	61.32	48.94	57.73	48.24	2.61x	4.5M
	HARP[53]	-	-	-	-	-	-	73.44	62.78	50.73	58.86	48.16	2.63x	
	TwinRep[24]	-	-	-	-	-	-	72.72	62.14	49.46	56.47	47.77	**2.80x**	
	EED(ours)	-	-	-	-	-	-	**74.10**	**64.66**	**52.42**	**60.92**	**51.83**	2.79x	

Table 5. Comparison of various AP methods and EED via sparsity rate on ResNet18 architecture.

	Setting	CIFAR-10 Clean	PGD	AA	C&W	DF	Speed up	SVHN Clean	PGD	AA	C&W	DF	Speed up	Prams
	AT[28]	87.05	56.14	48.02	57.60	53.10	1.00x	93.37	56.27	50.14	58.85	59.80	1.00x	11.2M
$s_r = 50\%$	R-ADMM[49]	82.14	54.10	46.23	56.27	45.80	1.36x	82.40	49.53	44.52	50.12	41.05	1.34x	5.6M
	HYDRA[35]	85.67	54.58	47.12	56.98	50.30	1.28x	91.12	53.47	45.83	50.27	46.58	1.26x	
	RST[11]	72.55	49.19	42.03	42.70	38.09	1.52x	78.12	42.09	34.62	52.07	38.14	1.54x	
	MAD[22]	77.33	53.21	44.45	57.21	43.12	1.37x	92.56	51.04	42.18	55.34	49.22	1.31x	
	Flying Bird[2]	83.02	53.18	45.13	57.33	48.02	1.24x	91.58	52.14	46.37	57.04	53.21	1.27x	
	HARP[53]	86.32	55.29	46.93	57.90	50.41	1.34x	92.13	53.29	47.52	57.26	54.08	1.36x	
	TwinRep[24]	81.45	54.13	48.45	57.82	48.27	1.30x	92.69	53.05	47.22	57.16	54.43	1.32x	
	EED(ours)	**88.07**	**57.83**	**52.35**	**57.92**	**53.12**	**1.64x**	**93.15**	**55.74**	**50.18**	**58.37**	**56.05**	**1.63x**	
$s_r = 70\%$	R-ADMM[49]	81.83	50.91	45.13	53.61	42.17	1.54x	78.24	50.32	45.21	51.36	42.28	1.55x	3.4M
	HYDRA[35]	84.21	53.40	46.03	56.40	47.29	1.56x	91.08	53.18	45.34	51.68	47.12	1.52x	
	RST[11]	63.89	42.67	34.31	35.05	31.52	**1.78x**	77.37	43.21	32.79	53.14	30.06	**1.79x**	
	MAD[22]	75.26	51.85	42.19	56.70	40.72	1.68x	91.25	51.37	43.11	56.42	45.33	1.63x	
	Flying Bird[2]	82.74	52.46	43.25	57.01	46.18	1.59x	91.62	52.09	47.25	57.12	50.27	1.63x	
	HARP[53]	85.40	54.71	45.10	**57.56**	48.35	1.65x	92.42	53.19	46.11	57.28	52.09	1.71x	
	TwinRep[24]	79.82	52.77	45.53	57.22	47.01	1.70x	92.36	53.27	47.13	56.49	51.43	1.69x	
	EED(ours)	**86.45**	**56.32**	**49.16**	57.51	**52.74**	1.77x	**92.85**	**55.16**	**50.32**	**58.68**	**56.24**	**1.79x**	
$s_r = 80\%$	R-ADMM[49]	81.25	48.00	43.92	49.17	39.11	1.68x	74.81	49.73	37.40	52.62	43.40	1.63x	2.2M
	HYDRA[35]	77.36	52.92	43.74	49.64	45.91	1.77x	91.06	52.22	47.62	55.13	46.13	1.74x	
	RST[11]	61.02	41.01	18.38	51.02	26.82	**1.86x**	82.39	46.29	36.35	52.65	39.27	1.83x	
	MAD[22]	74.18	50.38	41.27	54.17	38.44	1.69x	92.84	51.65	39.87	**59.80**	44.72	1.70x	
	Flying Bird[2]	81.07	51.62	44.41	56.08	45.29	1.79x	90.21	52.06	42.01	57.30	52.23	1.76x	
	HARP[53]	83.84	52.56	45.36	56.57	47.04	1.81x	92.60	54.16	45.89	57.28	51.24	1.80x	
	TwinRep[24]	77.26	52.04	43.52	56.60	46.18	1.82x	92.96	53.83	44.73	56.26	48.61	1.78x	
	EED(ours)	**86.13**	**55.71**	**48.13**	**57.03**	**51.97**	**1.86x**	**93.15**	**55.74**	**50.18**	58.37	**56.05**	**1.85x**	
$s_r = 90\%$	R-ADMM[49]	80.54	47.41	43.68	45.05	34.78	2.17x	84.34	51.91	38.40	49.46	42.87	2.23x	1.1M
	HYDRA[35]	76.74	47.42	43.34	44.81	44.68	2.49x	88.71	52.49	44.12	49.61	45.18	2.43x	
	RST[11]	60.92	38.24	14.31	26.98	25.19	**2.63x**	74.90	36.43	34.16	38.03	38.72	**2.68x**	
	MAD[22]	73.67	49.02	41.10	46.82	36.75	2.29x	89.42	44.29	37.46	48.90	41.13	2.31x	
	Flying Bird[2]	80.69	51.80	46.49	47.12	49.95	2.42x	91.60	**56.36**	39.81	**54.01**	52.41	2.36x	
	HARP[53]	**83.38**	50.41	45.40	47.75	48.53	2.33x	90.70	54.92	45.39	53.59	50.16	2.29x	
	TwinRep[24]	76.37	50.82	42.93	52.24	44.72	2.36x	88.90	53.06	46.71	50.22	48.34	2.38x	
	EED(ours)	83.26	**52.14**	**46.85**	**56.77**	**50.21**	2.51x	**91.74**	55.16	**47.32**	53.81	**53.57**	2.45x	
$s_r = 95\%$	R-ADMM[49]	71.42	42.31	42.56	39.91	31.41	2.94x	64.91	43.55	36.99	38.38	40.15	3.07x	0.6M
	HYDRA[35]	72.21	45.84	42.45	43.05	44.03	3.12x	85.71	50.56	45.29	45.53	46.92	3.19x	
	RST[11]	48.90	19.93	16.76	18.66	15.79	**3.58x**	61.55	25.90	23.94	26.39	28.58	**3.47x**	
	MAD[22]	58.90	41.29	38.96	41.47	37.08	2.82x	86.40	35.62	24.90	31.79	37.47	2.88x	
	Flying Bird[2]	75.40	45.63	**46.10**	42.68	44.79	3.10x	91.14	50.61	**47.43**	45.01	46.70	2.95x	
	HARP[53]	**77.13**	**50.41**	45.40	**47.75**	**48.53**	3.39x	**92.75**	**55.21**	45.95	51.3	**47.40**	3.19x	
	TwinRep[24]	64.97	47.58	41.47	43.92	45.03	3.46x	88.59	50.62	47.16	**52.31**	47.35	3.37x	
	EED(ours)	74.36	48.09	45.69	42.54	46.22	3.27x	90.76	52.74	47.19	47.23	46.99	3.30x	

When this value exceeds a certain threshold, it is considered that the prediction is unstable. For prediction confidence, we evaluated the confidence of the ensemble prediction at step t based on a specific threshold. The confidence calculation utilizes the maximum probability of the softmax distribution, which is described as:

$$Confidence_t = \max(softmax(\alpha_c^t))^2 \quad (30)$$

Here, softmax(α_c^t) represents the class probabilities, and the square of the highest probability indicates the confidence in the prediction. Ultimately, combining the above elements, q_t can be defined as:

$$q_t = \text{sig}(a \cdot Uncertainty_t + b \cdot Confidence_t) \quad (31)$$

In this manner, q_t is dynamically adjusted by incorporating factors such as prediction uncertainty and confidence, ensuring both efficiency and robustness.

B. Additional Experimental Results

B.1. Evaluation on Larger Datasets

Tab. 4 compares various AP methods and the proposed EED under a specific sparsity rate $s_r = 90\%$ condition, evaluated on the CIFAR-100 and ImageNet datasets. In this experiment based on the ResNet50 model, EED demonstrated either superior or comparable performance to existing AP methods across all major metrics. Notably, on the CIFAR-100 dataset, EED achieved a clean accuracy of 63.60%, showing the highest defense performance against various attacks, including PGD, AutoAttack (AA), and DeepFool. Furthermore, in the case of ImageNet, EED achieved a clean accuracy of 58.41%, with 30.54% defense against PGD and 26.89% defense against AA, demonstrating its outstanding robustness.

Additionally, EED maintained efficient computational speed while offering competitive performance compared to AP methods. EED recorded an average speed-up of 2.90x on CIFAR-100 and 2.84x on ImageNet, showing more efficient utilization of computational resources compared to previous methods. While HARP and TwinRep demonstrated competitive performance, EED outperformed them in terms of both robustness and efficiency. This highlights EED's ability to deliver high performance and computational efficiency even on large-scale datasets, underscoring its scalability and practical applicability.

B.2. Evaluation based on Sparsity

As discussed in Sec. 5.3, this experiment compares various AP methods with the proposed EED method via sparsity rates and then analyzes the performance on the CIFAR-10 and SVHN datasets using the ResNet18 architecture.

Table 6. Evaluation of EED with various AT methods on CIFAR-10 dataset when $s_r = 80\%$

	Setting	Clean	PGD	AA	C&W	DF	Speed up
Resnet18	Madry[28]	87.05	56.14	48.02	57.60	53.10	1.00x
	EED_{madry}	86.13	55.71	48.13	57.03	51.97	1.86x
	TRADES[52]	85.30	57.21	51.48	52.60	51.88	1.00x
	EED_{trades}	83.84	57.41	51.53	49.65	50.02	1.84x
	MART[42]	86.16	57.72	49.39	50.53	51.59	1.00x
	EED_{mart}	84.29	57.04	48.53	48.85	50.17	1.83x
VGG16	Madry[28]	82.70	54.49	48.52	54.91	56.91	1.00x
	EED_{madry}	81.39	54.26	47.49	53.27	53.94	1.79x
	TRADES[52]	83.18	55.72	49.09	54.59	57.89	1.00x
	EED_{trades}	82.13	54.79	47.26	51.98	56.30	1.80x
	MART[42]	77.44	57.51	46.20	51.38	59.56	1.00x
	EED_{mart}	77.45	57.14	46.16	51.44	57.55	1.77x

Table 7. Evaluation of EED via ensemble combiner on CIFAR-10 dataset when $s_r = 80\%$

	Setting	Clean	PGD	AA	C&W	DF	Speed up
Resnet18	EED_{avg}	**86.13**	**55.71**	48.13	**57.03**	51.97	1.86x
	EED_{max}	85.69	55.38	**48.52**	56.27	**52.58**	**1.88x**
VGG16	EED_{avg}	**81.39**	**54.26**	**47.49**	53.27	53.94	1.79x
	EED_{max}	80.83	54.11	46.70	**53.45**	**55.29**	**1.82x**

The primary comparison factors are the accuracy, robustness against various attacks, and the speed-up ratio, based on the model's sparsity rate (s_r).

According to Tab. 5, across sparsity rates ranging from $s_r = 50\%$ to $s_r = 95\%$, EED consistently outperforms other AP methods, with particular emphasis on clean accuracy and defense performance against PGD, AutoAttack, and DeepFool. These results suggest that EED maintains robustness even in sparsely compressed models through efficient and diverse ensemble strategies. At each sparsity rate, EED enhances robustness by utilizing differentiated pruning techniques and assigning diverse data subsets across AP-based sub-models, while improving efficiency through dynamic ensemble selection during the inference stage. Although performance is expected to decrease as s_r increases, EED maintains relatively high and stable performance compared to competing methods.

Notably, at a lower compression rate, i.e., $s_r = 50\%$, EED showed overall higher performance than not only other AP methods but also the pre-compressed AT model. For instance, on the CIFAR-10 dataset, EED outperformed AT by 1.69% in PGD and 4.33% in AA. Moreover, EED demonstrated significant inference speed-up, with a 1.64x speed-up on CIFAR-10 and a 1.63x speed-up on SVHN compared to other AP methods, reflecting a substantial improvement in computational efficiency. This performance boost is likely due to the impact of the DIE getting stronger as the

Table 8. Comparison of different ensemble sizes in EED on ResNet18.

	Setting	CIFAR-10						SVHN					
		Clean	PGD	AA	C&W	DF	Speed up	Clean	PGD	AA	C&W	DF	Speed up
$s_r = 50\%$	$\text{EED}_{d=1}$	84.53	53.27	45.62	55.43	45.18	1.61x	88.04	48.91	43.97	49.68	40.61	1.62x
	$\text{EED}_{d=3}$	85.14	54.29	46.85	56.37	50.04	1.63x	90.76	52.79	45.18	50.02	46.22	**1.63x**
	$\text{EED}_{d=4}$	**88.07**	57.83	52.35	**57.92**	**53.12**	1.64x	**93.15**	55.74	**50.18**	**58.37**	56.05	**1.63x**
	$\text{EED}_{d=5}$	87.43	**57.87**	**52.83**	57.19	52.68	**1.65x**	92.71	**55.98**	49.59	57.58	**56.41**	1.60x
	$\text{EED}_{d=6}$	86.24	56.21	51.89	56.12	52.81	1.59x	92.81	53.21	49.14	56.53	53.32	1.57x
$s_r = 80\%$	$\text{EED}_{d=1}$	81.57	53.42	45.74	55.71	45.24	1.74x	84.18	48.97	43.89	49.82	40.73	1.72x
	$\text{EED}_{d=3}$	84.22	54.11	46.59	56.22	49.88	1.78x	90.94	53.15	45.52	49.99	46.22	1.80x
	$\text{EED}_{d=4}$	**86.13**	**55.71**	**48.13**	**57.03**	**51.97**	**1.86x**	**93.15**	**55.74**	**50.18**	**58.37**	**56.05**	1.85x
	$\text{EED}_{d=5}$	84.87	55.29	47.92	56.78	51.53	1.83x	91.83	55.12	49.87	58.89	55.43	**1.86x**
	$\text{EED}_{d=6}$	82.44	54.03	46.72	55.95	50.29	1.84x	89.75	53.34	46.38	55.12	50.89	1.83x
$s_r = 90\%$	$\text{EED}_{d=1}$	79.77	48.81	42.93	51.96	45.48	2.36x	82.02	49.18	44.19	47.85	40.72	2.31x
	$\text{EED}_{d=3}$	81.33	51.28	**46.87**	56.59	49.01	2.44x	90.78	53.12	45.48	51.93	48.23	2.33x
	$\text{EED}_{d=4}$	**83.26**	**52.14**	46.85	**56.77**	**50.21**	**2.51x**	**91.74**	**55.16**	47.32	**53.81**	**53.57**	**2.45x**
	$\text{EED}_{d=5}$	82.12	51.92	46.67	56.52	49.89	2.47x	91.45	52.78	**47.41**	52.28	51.04	2.40x
	$\text{EED}_{d=6}$	80.95	50.18	44.32	53.48	47.77	2.46x	89.86	51.06	45.21	50.79	47.15	2.43x

ensemble size increases.

However, EED's performance decreased as the compression rate increased, primarily due to the reduction in ensemble diversity. Despite this, at $s_r = 90\%$, EED still outperformed most AP methods, and even at the extreme compression rate of $s_r = 95\%$, where methods like RST and MAD showed significant performance degradation, EED demonstrated robust defense performance with only minor performance decline.

B.3. Evaluation based on other AT methods

According to Tab. 6, EED, when combined with existing AT methods such as Madry-AT, TRADES-AT, and MART-AT, maintained similar performance across clean and various attacks (PGD, AA, C&W, DF), while achieving an average speed-up of 1.77x to 1.86x. Notably, in both ResNet18 and VGG16, EED successfully improved computational efficiency while minimizing performance degradation compared to the base model. For instance, EED_{MART}, compared to MART-AT, showed a slight improvement in clean accuracy on VGG16, and defense performance against attacks remains similar, while achieving a 1.77x inference speed-up. This indicates that EED has high compatibility with various AT methods and can be easily integrated with other AT approaches, providing further model compactness and better defense performance.

B.4. Average Combiner vs Max Combiner

As mentioned in Sec. 4.2, the final output of EED is computed using an average operation. In this experiment, we compared two ensemble output calculation methods: the average combiner $h(x) = \frac{1}{N}\sum_{i=1}^{N} h^i(x)$ and the maximum combiner $h_j(x) = \max_{i\in[N]}(h_j^i(x))$.

For ResNet18, the average combiner (EEDavg) showed relatively better performance in terms of clean accuracy, as well as defense against PGD and C&W attacks. However, the max combiner (EEDmax) demonstrated stronger defense against AA and DF attacks. In terms of speed-up, the max combiner achieved a slightly higher improvement of 1.88x. Similar trends were observed in VGG16, where the average combiner provided more stable performance in most cases, but the max combiner showed stronger defense against certain attacks (C&W and DF).

The results in Tab. 7 demonstrate that both combiners can enhance EED's performance while maintaining its compactness and efficiency. This suggests that both the average and max combiners extend the diversity and flexibility of EED, providing different advantages in specific attack scenarios.

B.5. Evaluation based on Ensemble Sizes

As mentioned in Sec. 4.1, we divide the training dataset for pruning into multiple subsets, and assign each subset to train a specific sub-model for the ensemble diversity. In the main paper, we set the number of subsets, d, to 4, and formed ensemble teams with 12 sub-models, denoted as $|EnsSet|$. Here, we evaluated the impact of the ensemble size on EED by adjusting d.

As shown in Tab. 8, when $d = 1$, i.e., when all sub-models are trained on the same dataset and the number of sub-models is small, we observed a general decline in clean accuracy and defense performance against attacks. This indicates that as the number of diverse sub-models in the ensemble decreases, the ensemble's diversity reduces, which in turn limits the ensemble's defensive capability. This performance degradation becomes more pronounced as the sparsity rate increases, which suggests that ensemble diversity becomes increasingly important as the model sparsifies.

On the other hand, as d increases, model diversity im-

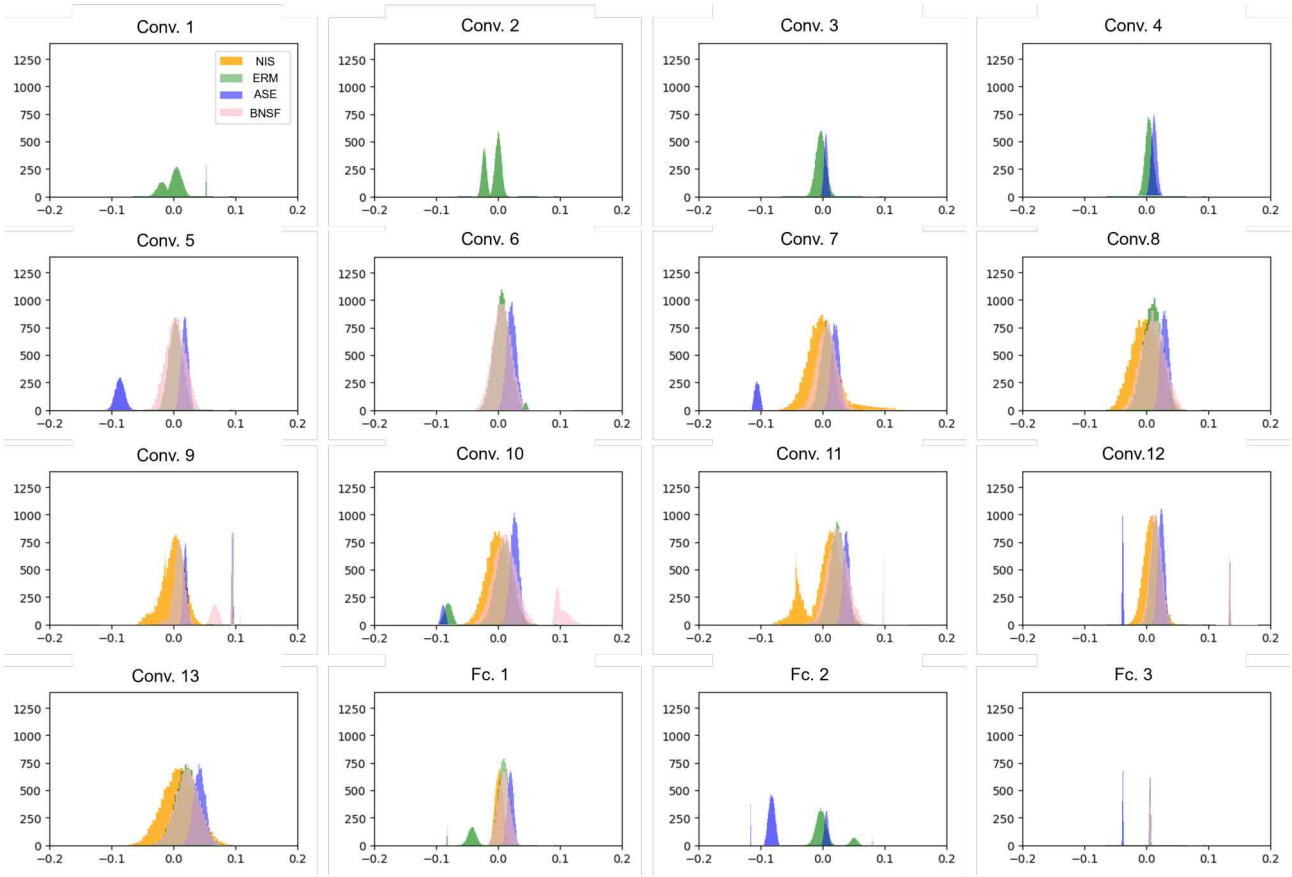

Figure 5. Parameter distribution in each layer of VGG16 pruned to 90% sparsity with AT.

proves, with $d = 4$ or $d = 5$ typically yielding the best performance. These configurations strike a balance between sufficient data sharing and the diversity of individual sub-models, which maximizes defense performance against attacks. However, when d was too large (e.g., $d = 6$), the amount of shared data between sub-models decreased, leading to insufficient training data for each sub-model. Consequently, model performance deteriorated, and defense capability weakened. This suggests that overly dividing the dataset to increase the number of sub-models may prevent individual models from learning essential core data, thereby weakening the overall defense capability of the ensemble.

Therefore, selecting an appropriate value for d plays a critical role in maximizing the defense performance and efficiency of EED, and it is necessary to maintain a balance between data diversity and the amount of shared data.

B.6. Analysis on Pruning Score via Parameter Distribution

We used multiple pruning importance scores to promote submodel diversity. In selecting the scores, we focused on their widespread use and effectiveness and their distinctiveness from each other. Fig.5 illustrates the distribution of parameters across different layers of an adversarially trained VGG16 model, which has been pruned to 90% sparsity on the CIFAR-10 dataset. The figure focuses on analyzing the effects of different pruning importance scores (NIS, ERM, ASE, BNSF). These distribution differences are closely related to the strategy employed in EED, where each importance score is used to maximize the model's diversity.

The parameter distribution varies significantly depending on the pruning importance score, with some criteria concentrating the distribution on specific layers, while others form broader and more diverse distributions. This indicates that each importance score emphasizes different patterns during training, contributing to the model's compressibility and defense efficiency.

These differences help the sub-models generated by applying different pruning scores in EED to achieve greater diversity and robustness. Since each sub-model in EED is trained using a different pruning importance score, the diversity of the overall model pool is ensured. This, in turn, mitigates the vulnerabilities of individual models and enhances the defense effectiveness.

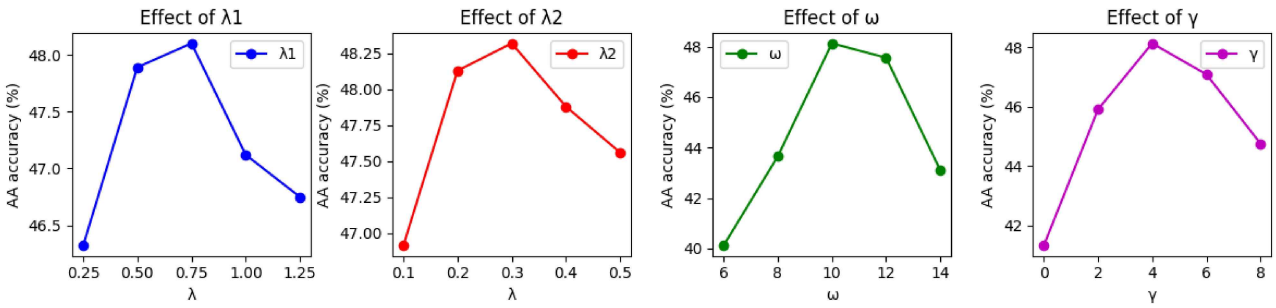

Figure 6. Effect of each hyperparameters in EED.

B.7. Analysis on Hyperparpameters

Fig.6 illustrates the effects of various hyperparameters on the accuracy of a classification model. The first graph shows the AA accuracy as a function of λ_1, reaching its peak at 0.6. The second graph depicts the effect of λ_2, where optimal performance is observed at 0.3. The third graph presents the accuracy corresponding to changes in ω, with a maximum accuracy achieved at 10. Lastly, the fourth graph illustrating the effect of γ shows the highest accuracy at 4. Collectively, these results emphasize the significance of hyperparameter tuning in optimizing model performance, which also indicate the importance of regularization terms.

C. Other Ensemble Defenses

In Sec. 4.1, we have discussed the diversity loss term (Div) and the regularization term (Reg) in existing ensemble defense techniques. In this section, we briefly describe four major ensemble-based adversarial defense methods, including ADP [31], GAL [18], DVERGE [48], and SoE [7], and evaluate the performance of EED, when applying the Div and Reg terms used in these methods, along with the DVERGE technique employed in the main paper.

C.1. ADP

ADP employs an ensemble approach through averaging, defined as $h(x) := \frac{1}{N}\sum_{i=1}^{N} h_i(x)$. The base classifiers are trained to minimize a loss function consisting of the cross-entropy loss for each classifier, a regularization term based on the Shannon entropy of the ensemble prediction, and a diversity loss that encourages various predictions. The loss function for the training example (x, y_x) is defined as:

$$\mathcal{L}_{\text{ADP}}(x, y_x) = \sum_{i=1}^{N} \ell_{\text{CE}}(h_i(x), y_x) - \alpha Reg(h(x)) + \beta \log Div(h_1(x), h_2(x), \ldots, h_N(x), y_x) \tag{32}$$

Here, the Shannon entropy is $\text{Reg}(p) = -\sum_{i=1}^{C} p_i \log(p_i)$. The diversity term $D(h_1, h_2, \ldots, h_N, y)$ measures the geometric diversity between the N distinct C-dimensional probability vectors. To compute the diversity, a normalized $(C-1)$-dimensional vector $\tilde{h}_i$ is derived by removing the element at the y-th position from h_i, and these vectors are stored as columns in the $(C-1) \times N$ matrix $\tilde{Reg}_y$. The diversity measure is computed as $\det(\tilde{Reg}_y^T \tilde{Reg}_y)$.

C.2. GAL

GAL proposes a loss function that considers both the prediction diversity and gradient diversity among the base classifiers. The loss function for the training example (x, y_x) is defined as follows:

$$\mathcal{L}_{\text{GAL}}(x, y_x) = \frac{1}{N}\sum_{i=1}^{N} D_{\text{KL}}(s_i(x) \| h_i(x)) - \alpha \log\left(\frac{2}{N(N-1)}\sum_{i=1}^{N}\sum_{j=i+1}^{N} \exp\left(JSD(s_i(x) \| s_j(x))\right)\right) + \frac{2\beta}{N(N-1)}\sum_{i=1}^{N}\sum_{j=i+1}^{N} \cos(\nabla D_{\text{KL}}(s_i(x) \| h_i(x)), \nabla D_{\text{KL}}(s_j(x) \| h_j(x))) \tag{33}$$

where $s_i(x)$ is the soft label vector obtained by label smoothing, and JSD is the Jensen-Shannon divergence.

C.3. DVERGE

DVERGE emphasizes the vulnerability diversity between the base classifiers to provide better adversarial robustness. For the i-th base classifier, the following is minimized:

$$\mathcal{L}_{\text{DVERGE}}(x, y_x) = \ell_{\text{CE}}(h_i(x), y_x) + \alpha \sum_{j \neq i} \mathbb{E}_{(x_s, y_{x_s}) \sim D, l \in [L]} \left[\ell_{\text{CE}}(h_i(\tilde{x}_{h_j(l)}(x, x_s)), y_{x_s})\right] \tag{34}$$

Here, $\tilde{x}_{h_j(l)}$ reflects non-robust features.

Table 9. Evaluation of EED with various ED losses on CIFAR-10 dataset when $s_r = 80\%$

	Setting	Clean	PGD	AA	C&W	DF	Speed up
Resnet18	EED_{ADP}	84.90	54.23	46.87	55.12	50.20	1.86x
	EED_{GAL}	85.30	54.70	47.50	55.75	50.90	1.85x
	$\text{EED}_{\text{DVERGE}}$	86.13	55.71	48.13	57.03	51.97	1.86x
	EED_{SoE}	85.85	55.10	47.90	56.40	52.10	1.82x
VGG16	EED_{ADP}	80.92	53.13	45.83	52.07	52.29	1.78x
	EED_{GAL}	81.05	53.79	46.38	52.52	53.07	1.80x
	$\text{EED}_{\text{DVERGE}}$	81.39	54.26	47.49	53.27	53.94	1.79x
	EED_{SoE}	80.59	53.95	46.10	53.11	54.58	1.81x

C.4. SoE

SoE is based on vulnerability diversity and two training phases utilizing different adversarial examples. The loss function for the i-th base classifier h_i is defined as:

$$\begin{aligned}\mathcal{L}_{\text{SoE}}(x, y_x) = &\sum_{j=1}^{N} \ell_{\text{BCE}}(h_j(y_x, \tilde{x}_i), g_j(\tilde{x}_i)) \\ &-\sigma \ln \sum_{j=1}^{N} \exp\left(-\frac{\ell_{\text{CE}}(h_j(\tilde{x}_i), y_x)}{\sigma}\right)\end{aligned} \tag{35}$$

C.5. Analysis on Ensemble Defenses

We evaluated EED with various ensemble loss functions on the CIFAR-10 dataset. The results shown in Tab.9 demonstrate that EED has strong compatibility with a range of ensemble defense strategies, delivering notable improvements in adversarial robustness and inference speed. EED consistently showed excellent performance against attack types such as PGD, AA, C&W, and DF, with particularly high robustness observed when using vulnerability diversity terms like those in DVERGE and SoE, which performed well across all attack types.

D. Limitations and Future Work

The proposed Efficient Ensemble Defense (EED) enhances the diversity of compressed models, maintains robustness against adversarial attacks, and effectively reduces model capacity. However, EED has a few limitations yet.

First, the training cost is relatively high. The process of generating sub-models using multiple pruning metrics and combining them into an ensemble can increase training time compared to conventional Adversarial Pruning (AP). This is an inevitable aspect of ensemble defense, but when compared to ensembles using multiple base models, the cost is significantly lower. Unlike AP, which requires a lot of cost for fine-tuning, EED can address this by leveraging the ensemble, thus mitigating the overhead during sub-model generation. However, when using large-scale datasets or complex models, it can still be difficult to train models in environments with limited computing resources.

Additionally, when applying extreme sparsity, as shown in Sec. 5.3 and Tab.5, there is a tendency for model performance to degrade. Although EED aims to use compressed models for efficiency, applying excessively high sparsity can lead to performance degradation. This happens because, as sparsity increases, ensemble diversity decreases, which diminishes the advantages of the ensemble approach.

Future research could explore several approaches to overcome these limitations. First, optimization methods for reducing training costs should be considered. In particular, efficient algorithms are required to reduce the computational overhead in the sub-model creation and ensemble combination process. For example, techniques that optimize parameter sharing during training or streamline the sub-model selection process could help shorten training times. Next, more refined pruning techniques need to be developed to address the performance degradation caused by high sparsity. Current pruning methods may damage the model's structure and important parameters as sparsity increases, so exploring ways to maximize compression while minimizing performance loss is crucial. Additionally, techniques that maintain diversity among sub-models, even with high compression, could be developed.