Grow to Compress? Efficient Training of Robust Networks on the Edge

Vignesh Sundaresha & Naresh Shanbhag
University of Illinois at Urbana-Champaign
Urbana, USA
{vs49, shanbhag}@illinois.edu

Abstract

The ubiquitous deployment of deep learning systems on resource-constrained Edge devices is hindered by their high computational complexity coupled with their fragility to out-of-distribution (OOD) data, especially to naturally occurring common corruptions. Current solutions rely on the Cloud to train and compress models before deploying to the Edge. This incurs high energy and latency costs in transmitting locally acquired field data to the Cloud while also raising privacy concerns. We propose GEARnn (Growing Efficient, Accurate, and Robust neural networks) to grow and train robust networks in-situ, i.e., completely on the Edge device. Starting with a low-complexity initial backbone network, GEARnn employs One-Shot Growth (OSG) to grow a network satisfying the memory constraints of the Edge device using clean data, and robustifies the network using Efficient Robust Augmentation (ERA) to obtain the final network. We demonstrate results on a NVIDIA Jetson Xavier NX, and analyze the trade-offs between accuracy, robustness, model size, energy consumption, and training time. Our results demonstrate the construction of efficient, accurate, and robust networks entirely on an Edge device.

1 Introduction

The ubiquitous practical deployment of deep neural networks is mainly hindered by their lack of robustness and high computational cost. Prior art has shown that these deep networks are extremely fragile to adversarial perturbations [31][8] and out-of-distribution (OOD) data [12][22]. Natural corruptions [12] (a specific type of OOD data) are more commonly encountered at the Edge where real-time data is being continually acquired, e.g., video sequences acquired by on-board cameras in autonomous agents (self-driving cars, field robots, drones), which tend to be distorted by weather and blur. The state-of-the-art defense against these corruptions employs robust data augmentation [13, 11, 23] which incurs a huge computational cost when implemented on an Edge device. Fig. 1 indicates that it takes more than 2 days to robustly train a VGG-19 network [29] on a simple CIFAR-10 dataset when implemented on NVIDIA Jetson Xavier [24] Edge device. Even for a small 5% VGG-19 network it takes more than a day, thus highlighting the non-trivial nature of the problem. This is a huge concern because Edge devices are typically battery-powered and such large training costs reduce their operational life-time.

Traditional solutions for network compression such as pruning [9, 19, 4], quantization [26, 16] and neural architecture search (NAS) [21, 37] mainly target Edge inference, and are not suited for Edge training since they start with hard-to-fit over-parameterized networks that require the large computational resources of the Cloud. However, transmitting local data to the Cloud incurs energy and latency costs while also raising privacy concerns, thus requiring training to happen fully on the Edge. Hence we ask the question: *Is it possible to design and train compressed robust networks fully on the Edge?* Our proposed solution GEARnn (Growing Efficient, Accurate, and Robust neural networks) is based on the family of growth algorithms [2, 33, 5, 35] that gradually increase the size

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

of an initial backbone network to reach the robust accuracy of a full network but at a fraction of its size, training complexity, and energy consumption.

Prior work on network growth [33, 34, 36] do not consider robustness to common corruptions since they use clean data during training, while works that consider robustness train fixed-sized networks using augmented data [13, 23] without considering the efficiency of robust training. Hence, in order to grow robust networks efficiently on the Edge, we ask the following questions: **Q1)** *are compressed networks designed using growth susceptible to common corruptions?* **Q2)** *how to construct growth algorithms that maximize training efficiency when designing robust networks?* **Q3)** *what are the network topologies generated when growth algorithms design compressed networks?* We answer these questions by proposing our method GEARnn to efficiently grow robust networks.

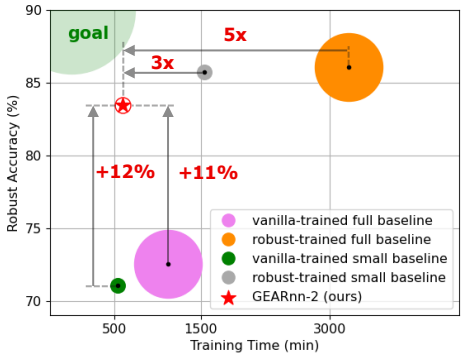


Figure 1: Improvements in robust accuracy, training time, and model size (area of circles) of our proposed GEARnn method measured on NVIDIA Jetson Xavier NX [24] Edge device. Robust accuracy is evaluated on CIFAR-10-C for GEARnn, full network baselines (VGG-19), and small network baselines (5% VGG-19 networks with same topology as GEARnn-2). For robust training, we employ AugMix [13]. GEARnn demonstrates significant reduction in training complexity over robust baselines at similar robust and clean (shown in Section 6.4) accuracy.

Contributions: We make the following contributions (Fig. 2):

1. To the best of our knowledge, our work is the first to grow networks robust to common corruptions. Additionally, we also consider the training efficiency of our methods.
2. We answer **Q2** by proposing GEARnn (Growing Efficient Accurate and Robust neural networks) algorithm (see Fig. 2) that combines One-Shot Growth (OSG) and Efficient Robust Augmentation (ERA).
3. We answer **Q3** by showing that vanilla CNNs have relatively higher output channels in the initial layers while residual CNNs have a steady zigzag pattern (Section 6.3).
4. We show that GEARnn generated networks shine on all four metrics simultaneously - clean accuracy, robust accuracy, training efficiency and inference efficiency by implementing them on a real-life Edge device – the NVIDIA Jetson Xavier NX (Section 6.4).

2 Background and Related Work

Robust Data Augmentation: This is the most commonly used method for addressing corruptions due to its ease of integration into the training flow and ability to replicate low-level structural distortions. AugMix [13], PRIME [23] and FourierMix [30] combine chains of stochastic image transforms and enforce consistency using a suitable loss function to generate an augmented sample from a clean image. DeepAugment [11] randomly distorts the parameters of an image-to-image network to generate augmented images. CARDs [4] combines data augmentation [13] and pruning [6] to find compact robust networks embedded in large over-parameterized networks. Unlike our proposed GEARnn algorithm, all these techniques significantly increase the complexity over vanilla training, and are thus inappropriate for Edge deployment.

Growth Techniques: A typical growth algorithm starts with a small initial backbone model whose size is gradually increased until the desired performance or network topology is reached. Neural network growth has been previously used in optimization [7], continual learning [27, 17] and in speeding up the training of large networks [2]. Recent works [5, 36] look at improving growth training dynamics, growing the width [34], depth [32] or both [33, 35]. However, none of these methods address the issue of robustness to common corruptions or demonstrate the utility for training on a resource-constrained Edge setting, which is our focus. Though our work GEARnn builds upon Firefly [33], it is flexible and can incorporate other growth methods mentioned above.

3 Notation and Problem Setup

Notation: Let $f : \mathbb{R}^d \to [C]$ be a hard classifier which classifies input $\mathbf{x} \in \mathbb{R}^d$ into one of C classes. We choose f to be a convolutional neural network (CNN) with L layers (depth) and $\{w_l\}_{l=1}^{L}$ output channels (widths). The network f is trained on n samples $(\mathbf{x}, y) \sim \mathcal{D}_{\text{in}}$, where $(\mathbf{x}, y) \in \mathbb{R}^d \times [C]$ and $\mathcal{D}_{\text{in}}$ denotes the "in-distribution" or "clean" data. We denote $n_C = \frac{n}{C}$ as the number of samples per class. $\mathcal{L}_{\text{CE}}$ represents the cross-entropy loss and $\mathcal{L}_{\text{aug}} = \mathcal{L}_{\text{CE}} + \lambda\mathcal{L}_{\text{JSD}}$ represents the augmentation loss where $\mathcal{L}_{\text{JSD}}$ is the Jensen-Shannon divergence loss described in [13].

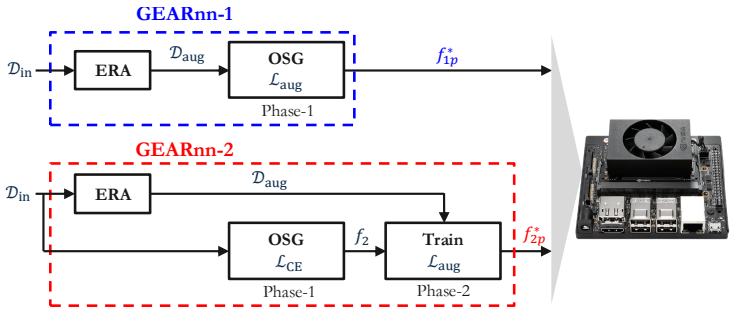


Figure 2: Proposed approach: GEARnn-1 performs One-Shot Growth (OSG) on augmented data ($\mathcal{D}_{\text{aug}}$) generated by Efficient Robust Augmentation (ERA) using clean data ($\mathcal{D}_{\text{in}}$) in a single phase (1-Phase). GEARnn-2 performs OSG using $\mathcal{D}_{\text{in}}$ first followed by parametric training on $\mathcal{D}_{\text{aug}}$ in two consecutive phases (2-phase). Here $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{aug}}$ denote the cross-entropy loss and augmented loss, respectively.

During inference, f can be exposed to samples from both $\mathcal{D}_{\text{in}}$ and $\mathcal{D}_{\text{out}}$ ("out-of-distribution" or "corrupted" data). In case of common corruptions, $(\mathbf{x}_{\text{out}}, y) \sim \mathcal{D}_{\text{out}}$ is obtained by $\mathbf{x}_{\text{out}} = \kappa(\mathbf{x}_{\text{in}}, s)$, where $(\mathbf{x}_{\text{in}}, y) \sim \mathcal{D}_{\text{in}}$, κ is a corruption filter and s is the severity level of the corruption. We denote $p_e = \Pr(\hat{y} \neq y)$ as the classification error at inference where $\hat{y} = f(\mathbf{x}_{\text{test}})$. When $(\mathbf{x}_{\text{test}}, y) \sim \mathcal{D}_{\text{in}}$, we define $(1 - p_e)$ as clean accuracy $\mathcal{A}_{\text{cln}}$, and when $(\mathbf{x}_{\text{test}}, y) \sim \mathcal{D}_{\text{out}}$, we define $(1 - p_e)$ as robust accuracy $\mathcal{A}_{\text{rob}}$. The value of p_e is determined empirically in this work.

Problem: Our primary objective is to maximize the empirical clean and robust accuracies ($\mathcal{A}_{\text{cln}}$ and $\mathcal{A}_{\text{rob}}$) while ensuring the network complexity ($\sum_{l=1}^{L} w_l$) is small. Along with these two criteria, we also prioritize reduction in training time (t_{tr}) and training energy consumption (E) on hardware.

4 Growing Efficient Accurate and Robust Neural Networks (GEARnn)

As shown in Fig. 2, two flavors of GEARnn algorithms are proposed – GEARnn-1 and GEARnn-2. While GEARnn-1 leverages the 1-Phase (joint growth and robust training) training, GEARnn-2 employs the 2-Phase (sequential growth and robust training) approach. Both flavors incorporate One-Shot Growth (OSG) and Efficient Robust Augmentation (ERA) in different ways. In this section, we describe OSG and ERA. The full description of the GEARnn algorithms can be found in Appendix F.

4.1 One-Shot Growth (OSG)

One-Shot Growth (OSG) employs labeled data to perform a single growth step sandwiched between two training stages. The initial backbone f_0 is first trained for $\mathcal{E}_1$ epochs. The resulting network f_1 is grown over $\mathcal{E}_g$ epochs to obtain the grown network f_g, i.e., $f_g = \mathcal{G}(f_1|\gamma, \mathcal{D}, \mathcal{L}, \mathcal{E}_g)$, where $\mathcal{G}$ is the growth technique which is nominally Firefly [33] in our work. The final network f_2 is obtained by training f_g over $\mathcal{E}_2$ epochs.

The growth technique $\mathcal{G}$ is described below:

$$\begin{aligned} f_g = \arg\min_{f} \quad & \mathcal{L}(f, \mathcal{D}|f_1) \\ \text{s.t.} \quad & f \in \partial(f_1, \epsilon) \\ & \mathcal{C}(f) \leq (1+\gamma)\,\mathcal{C}(f_1) \end{aligned} \tag{1}$$

where $\partial(f_1, \epsilon)$ represents the growth neighbourhood for topology search, $\mathcal{C}(f) = \sum_{l=1}^{L} w_l$ represents the complexity estimate of network f and γ denotes the growth ratio. The neighbourhood $\partial(f_1, \epsilon)$ is expanded in two ways - splitting and growing new neurons - as described in [33, 34].

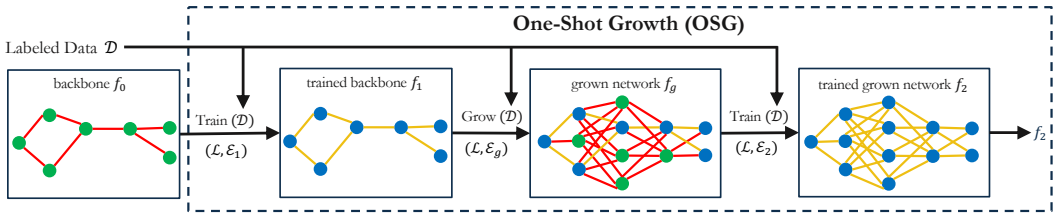


Figure 3: OSG takes in labeled data ($\mathcal{D}$) and backbone network f_0, and performs a training step, a growth step, and a training step in sequence to generate network f_2. The 2-tuple $(\mathcal{L}, \mathcal{E}) =$ (loss function, number of epochs) employed in each step.

4.2 Efficient Robust Augmentation (ERA)

Efficient Robust Augmentation (ERA) employs clean data ($\mathcal{D}_{\text{in}}$) to generate augmented data ($\mathcal{D}_{\text{aug}}$) in an efficient manner. The clean sample $\mathbf{x}$ (where $(\mathbf{x}, y) \sim \mathcal{D}_{\text{in}}$) is passed through a set of transforms $a_1, a_2, ..., a_{d_j}$ to obtain the transformed sample $A_j(x)$, which is then combined linearly with the clean sample to give the augmented sample $\mathbf{x}_j^{\text{aug}}$. We concatenate $(J-1)$ such augmented samples $\{\mathbf{x}_j^{\text{aug}}\}_{j=1}^{J-1}$ along with the clean sample to obtain our Efficient Robust Augmentation $\mathcal{R}((\mathbf{x}, y)|\mathcal{T})$.

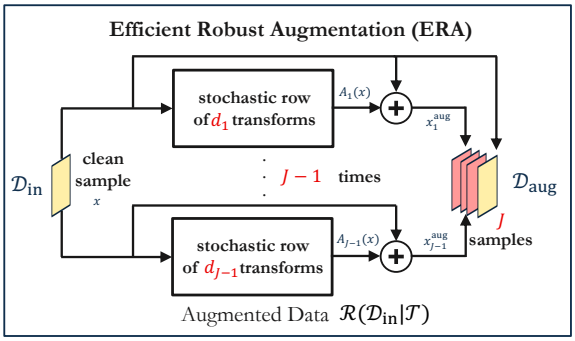


Figure 4: ERA takes in clean data ($\mathcal{D}_{\text{in}}$) as input and applies a set of stochastic transforms to generate augmented data ($\mathcal{D}_{\text{aug}}$).

$$
\begin{aligned}
A_j(\mathbf{x}) &= a_1 \circ a_2 \circ ... \circ a_{d_j}(\mathbf{x}) \\
\mathbf{x}_j^{\text{aug}} &= p\mathbf{x} + (1-p)A_j(\mathbf{x}) \\
\mathcal{R}((\mathbf{x}, y)|\mathcal{T}) = (\{\mathbf{x}_1^{\text{aug}}, ..., \mathbf{x}_{J-1}^{\text{aug}}, \mathbf{x}\}, y) &\implies \mathcal{D}_{\text{aug}} := \mathcal{R}(\mathcal{D}_{\text{in}}|\mathcal{T}) \\
\text{where } a_i \sim \text{Unif}(\mathcal{T}),\ p \sim \beta(1,1),\ d_j \sim \text{Unif}(\{1, ..., D\}),\ j &\in \{1, ..., J-1\}
\end{aligned}
\tag{2}
$$

where $\mathcal{T}$ denotes the set of transforms (from AugMix [13]), β and Unif represent the beta and uniform distributions, respectively. The width (W), depth (D) and JSD samples (J) for the stochastic transforms in the augmentation are chosen as $(W, D, J) = (1, 3, 4)$. In GEARnn-2, grown network f_2 (see Fig. 3) obtained using clean data OSG is trained for $\mathcal{E}_r$ epochs using $\mathcal{D}_{\text{aug}}$ generated by ERA.

5 Experimental Setup

For comprehensive details of the experimental setup, please refer to Appendix A.

Hardware: For the server-based experiments, we use one NVIDIA Quadro RTX 6000 GPU ("Quadro"). For the Edge-based experiments, we use the NVIDIA Jetson Xavier NX [24] ("Jetson").

Table 1: GEARnn hyperparameters for different networks and datasets.

Dataset	γ			Small	GEARnn-1		GEARnn-2		
	Mob.	VGG	Res.	$\mathcal{E}$	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_r$
CIFAR-10	1.8	0.9	0.6	160	40	40	40	40	40
CIFAR-100	2.0	1.5	0.8	160	50	50	40	40	50
Tiny ImageNet	2.0	1.5	0.8	160	50	50	40	40	50

Metrics: Clean accuracy $\mathcal{A}_{\text{cln}}(\%)$, robust accuracy $\mathcal{A}_{\text{rob}}(\%)$, number of floating-point parameters (Size ($\%$)), wall-clock training time t_{tr} (in minutes), per-sample wall-clock inference time t_{inf} (in seconds) and energy consumption E (in Joule) are used as the metrics.

Baselines: In the absence of prior work on robust growth, we propose our own baselines Small ($\mathcal{D}_{\text{in}}$) and Small ($\mathcal{D}_{\text{aug}}$), both of which use 160 training epochs to be consistent with [4]. They are networks with the same size and topology as the final GEARnn-2 network trained with random initialization on clean data and augmented data, respectively. Their corresponding full (original) network versions are called Full ($\mathcal{D}_{\text{in}}$) and Full ($\mathcal{D}_{\text{aug}}$). The rationale for this baseline choice is explained in Appendix A.

6 Main Results

6.1 Q1: Corruption Robustness of Growth Networks

Table 2 compares different growth methods with the full network when trained for 160 epochs on clean CIFAR-10 data using VGG-19. We find that even though growth networks have $\sim$5% size of the full VGG-19 model, they are able to achieve similar accuracy and robustness but at a much lower training and inference cost. We observe that the accuracies drop and training costs increase as the growth steps rise, thus stopping us from going beyond 4 growth steps. Table 2 shows why compression techniques like *pruning or quantization are not appropriate baselines for our work* since they require expensive training of the full network to be done on the Edge. Unlike what was found in [14, 20] for pruning, we find that the compact networks obtained by growth have similar robustness to the full network. However, there is still a drop in their robust accuracy compared to clean accuracy, and in the next section we look at how to robustify these growth networks using data augmentation.

Table 2: Comparing robustness and training cost for different growth methods and the full network for VGG-19 and CIFAR-10 on Quadro.

Growth Steps	Size(%)	$\mathcal{A}_{\text{cln}}(\%)$	$\mathcal{A}_{\text{rob}}(\%)$	t_{tr} (min)	E (kJ)
1	5.5	92.14	70.04	**23**	163
2	5.5	92.18	70.22	27	**138**
3	**5.3**	91.84	69.84	28	145
4	**5.3**	91.91	70.07	31	224
Full ($\mathcal{D}_{\text{in}}$)	100	**93.06**	**72.36**	45	469

6.2 Q2: One-Shot vs. Multi-Shot Growth & 1-Phase vs. 2-Phase

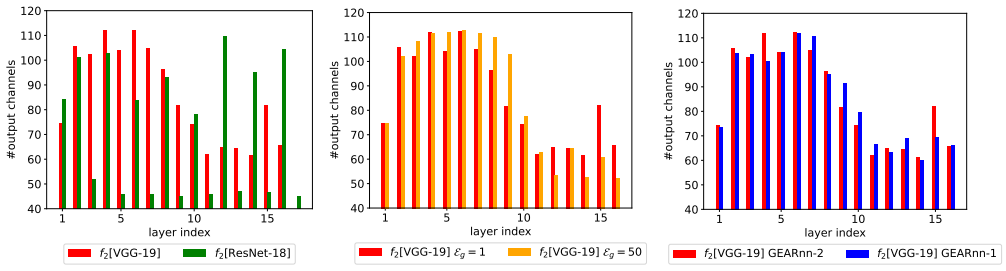
Since GEARnn employs OSG (One-Shot Growth) for growing networks, it begs the question if we are missing anything if multiple growth steps (m-Shot Growth) were to be permitted. To answer this question, we compare the clean and robust accuracies along with training time and energy for different growth steps in Table 3. All m-Shot Growth methods start with the same initial backbone f_0 (1.4% of full model size) and perform growth to reach f_2 (5% of full model size) using different growth ratios.

Table 3 indicates that OSG is comparable or better than the other m-Shot Growth methods in all the metrics. This result can be attributed to the lower training overhead of growth stage in OSG compared to the m-Shot Growth methods. For each growth step, GEARnn-2 is better than the corresponding GEARnn-1 solution on all the metrics (and red highlights the best solution across the table). Thus, Table 3 clearly highlights that 2-Phase approach using One-Shot Growth is the best combination to grow robust networks efficiently on the Edge.

Table 3: Comparison of training complexities and accuracies for different growth methods implemented for VGG-19 and CIFAR-10 on Jetson with 80 epochs during growth.

Growth Steps	GEARnn-1				GEARnn-2			
	$\mathcal{A}_{\text{cln}}(\%)$	$\mathcal{A}_{\text{rob}}(\%)$	t_{tr} (min)	E (kJ)	$\mathcal{A}_{\text{cln}}(\%)$	$\mathcal{A}_{\text{rob}}(\%)$	t_{tr} (min)	E (kJ)
1	**90.94**	**82.25**	652	207	**92.07**	**83.45**	596	**155**
2	90.01	81.92	**640**	**191**	91.94	83.34	**593**	157
3	89.73	80.86	653	194	91.79	83.05	624	177
4	89.90	81.08	845	223	91.65	82.75	645	173

6.3 Q3: Network Topologies generated by Growth



(a) (b) (c)

Figure 5: Average output channels vs. layer index for CIFAR-10 on Quadro is shown. Plot (a) looks at the impact of network architecture and highlights the non-uniform growth pattern in plain CNNs versus steady zigzag pattern in residual CNNs. Plots (b) and (c) indicate that modifying the number of growth epochs ($\mathcal{E}_g$) or performing 1-Phase robust growth does not affect the topology pattern much.

In this section, we look at the growth topology patterns ($\{w_l\}_{l=1}^{L}$) as a function of layer index l. Specifically, we investigate these patterns in the simple setting of OSG ($\mathcal{D}_{\text{in}}$) implemented on

Table 4: Comparison of accuracy, robustness, and efficiency between the baselines and GEARnn for VGG-19 using CIFAR-10, CIFAR-100 and Tiny ImageNet on Quadro. See Fig. 6 for robustness comparison between Small ($\mathcal{D}_{\text{aug}}$) and GEARnn-2 at similar training cost.

Architecture		CIFAR-10					CIFAR-100					Tiny ImageNet				
(full model size)	Method	Size (%)	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Training t_{tr}(min)	Training E(kJ)	Size (%)	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Training t_{tr}(min)	Training E(kJ)	Size (%)	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Training t_{tr}(min)	Training E(kJ)
VGG-19 (20M)	Small ($\mathcal{D}_{\text{in}}$)	5	92.69	70.57↓	31	241	9	68.07	41.24↓	38	335	9	53.9	17.78↓	218	2040
	Full ($\mathcal{D}_{\text{in}}$)	100	93.06	72.36↓	45	469	100	69.97	42.49↓	46	433	100	56.89	21.46↓	274	2740
	Small ($\mathcal{D}_{\text{aug}}$)	**5**	93.08	85.73	215	1140	**9**	70.01	56.94	219	927	**9**	55.51	30.01	668	7120
	Full ($\mathcal{D}_{\text{aug}}$)	100	**93.43**	**86.50**	197	950	100	**70.39**	**58.14**	194	985	100	54.40	**31.32**	795	9460
	GEARnn-1	**5**	91.25	82.86	86	552	**9**	65.73	52.68	111	779	**9**	54.38	28.56	428	4220
	GEARnn-2	**5**	92.18	83.77	**53**	**298**	**9**	68.44	54.31	**65**	**566**	**9**	**56.19**	29.79	**357**	**3219**

Table 5: Comparison of accuracy, robustness, inference efficiency, and training efficiency between the baselines and GEARnn for CIFAR-10 and CIFAR-100 using VGG-19 on Jetson. Due to computational limitations, the results for Tiny ImageNet are excluded for Jetson.

Method	CIFAR-10						CIFAR-100					
	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Inference Size%	Inference t_{inf}(ms)	Training t_{tr}(min)	Training E(kJ)	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Inference Size%	Inference t_{inf}(ms)	Training t_{tr}(min)	Training E(kJ)
Small ($\mathcal{D}_{\text{in}}$)	92.97	71.08↓	5	1.0	533	128	67.92	40.49↓	9	1.4	714	187
Full ($\mathcal{D}_{\text{in}}$)	92.32	72.53↓	100	2.4	1124	268	69.69	43.01↓	100	2.4	1117	309
Small ($\mathcal{D}_{\text{aug}}$)	**93.36**	85.73	**5**	**1.0**	1543	522	**70.07**	56.68	**9**	**1.4**	2016	678
Full ($\mathcal{D}_{\text{aug}}$)	92.92	**86.06**	100	2.4	3219	937	68.81	**57.86**	100	2.4	3199	1043
GEARnn-1	90.94	82.25	**5**	1.2	652	207	62.89	49.63	**9**	1.5	936	**281**
GEARnn-2	92.07	83.45	**5**	**1.0**	**596**	**155**	67.59	53.64	**9**	**1.4**	**884**	328

CIFAR-10 for $(\mathcal{E}_1, \mathcal{E}_2)$ = (40, 40) and an initial backbone f_0 with $\{w_l\}_{l=1}^{L} = 45$. The bar plots represent the mean width ($\mathbb{E}[w_l]$) across four random seeds.

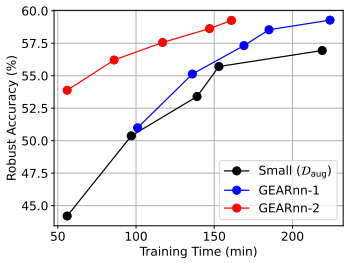
Backbone architecture: For plain CNNs like VGG-19 [29] - the initial layers have higher number of convolutional filters compared to the final layers. This correlates well with the observations seen in quantization [28] where the initial layers require higher precision compared to the final layers. However, in case of residual networks like ResNet-18, the pattern is oscillating as shown in Fig. 5a. The invariance in depth can be attributed to the direct gradient flow facilitated by the shortcut connections which makes each residual block act independently of the depth.

Data and Growth Epochs: The above experiments were performed for a single growth epoch ($\mathcal{E}_g$ = 1) on clean data. The effect of $\mathcal{E}_g = 50$ and using ERA data for growth (GEARnn-1) is shown in Fig. 5b and Fig. 5c. The topology pattern in both cases remains similar to OSG ($\mathcal{D}_{\text{in}}$) $\mathcal{E}_g = 1$.

6.4 Results across Network Architectures, Datasets and Devices

Table 4 and Table 5 show that GEARnn-2 achieves comparable accuracies while being consistently better in terms of training time and training energy consumption over the robust baselines for VGG-19 (other network architecture results are in Appendix B) across all datasets and devices. Specifically, the average reduction in training time (energy consumption) is $\mathbf{3.0}\times$ ($\mathbf{2.6}\times$) for Quadro and $\mathbf{4.5}\times$ ($\mathbf{4.6}\times$) for Jetson compared to Full ($\mathcal{D}_{\text{aug}}$). Furthermore, we find GEARnn-1 is inferior to GEARnn-2 on all the four metrics thereby indicating the efficacy of the 2-Phase approach.

A key reason underlying GEARnn-2's training efficiency is the reduction in the number of robust training epochs $\mathcal{E}_r$ made possible by the OSG initialization in Phase-1. Fig. 6 shows that for the same training time, GEARnn-2 provides better robustness than Small ($\mathcal{D}_{\text{aug}}$) and GEARnn-1. Similar results obtained for CIFAR-10 and other network architectures is shown in Appendix E.



(a)

Figure 6: GEARnn-2 achieves higher robustness at the same training time, for VGG-19/CIFAR-100 on Quadro.

7 Conclusion

We addressed the problem of growing robust networks efficiently on Edge devices. Specifically, we concluded that a 2-Phase approach with distinct clean growth and robust training phases is significantly more efficient than a 1-Phase approach which employs augmented data for growth. We encapsulated this result into the GEARnn algorithm and experimentally demonstrated its benefits on a real-life Edge device. An interesting and non-trivial extension of our work would be to use unlabeled data for growing efficient and robust networks.

References

[1] Bonghi, R.: Jetson-stats. https://pypi.org/project/jetson-stats/

[2] Chen, T., Goodfellow, I., Shlens, J.: Net2net: Accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641 (2015)

[3] Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., Hein, M.: Robustbench: a standardized adversarial robustness benchmark. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2021), https://openreview.net/forum?id=SSKZPJCt7B

[4] Diffenderfer, J., Bartoldson, B., Chaganti, S., Zhang, J., Kailkhura, B.: A winning hand: Compressing deep networks can improve out-of-distribution robustness. Advances in neural information processing systems **34**, 664–676 (2021)

[5] Evci, U., van Merrienboer, B., Unterthiner, T., Vladymyrov, M., Pedregosa, F.: Gradmax: Growing neural networks using gradient information. arXiv preprint arXiv:2201.05125 (2022)

[6] Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018)

[7] Fukumizu, K., Amari, S.i.: Local minima and plateaus in hierarchical structures of multilayer perceptrons. Neural networks **13**(3), 317–327 (2000)

[8] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)

[9] Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015)

[10] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[11] Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al.: The many faces of robustness: A critical analysis of out-of-distribution generalization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8340–8349 (2021)

[12] Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261 (2019)

[13] Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019)

[14] Hooker, S., Courville, A., Clark, G., Dauphin, Y., Frome, A.: What do compressed deep neural networks forget? arXiv preprint arXiv:1911.05248 (2019)

[15] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

[16] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. Advances in neural information processing systems **29** (2016)

[17] Hung, C.Y., Tu, C.H., Wu, C.E., Chen, C.H., Chan, Y.M., Chen, C.S.: Compacting, picking and growing for unforgetting continual learning. Advances in Neural Information Processing Systems **32** (2019)

[18] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[19] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)

[20] Liebenwein, L., Baykal, C., Carter, B., Gifford, D., Rus, D.: Lost in pruning: The effects of pruning neural networks beyond test accuracy. Proceedings of Machine Learning and Systems **3**, 93–138 (2021)

[21] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European conference on computer vision (ECCV). pp. 19–34 (2018)

[22] Mintun, E., Kirillov, A., Xie, S.: On interaction between augmentations and corruptions in natural corruption robustness. Advances in Neural Information Processing Systems **34**, 3571–3583 (2021)

[23] Modas, A., Rade, R., Ortiz-Jiménez, G., Moosavi-Dezfooli, S.M., Frossard, P.: Prime: A few primitives can boost robustness to common corruptions. In: European Conference on Computer Vision. pp. 623–640. Springer (2022)

[24] NVIDIA, C.: Nvidia jetson xavier nx for embedded amp; edge systems. https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-xavier-nx/

[25] NVIDIA, C.: System management interface smi. https://developer.nvidia.com/nvidia-system-management-interface

[26] Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: European conference on computer vision. pp. 525–542. Springer (2016)

[27] Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)

[28] Sakr, C., Shanbhag, N.: An analytical method to determine minimum per-layer precision of deep neural networks. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1090–1094. IEEE (2018)

[29] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[30] Sun, J., Mehra, A., Kailkhura, B., Chen, P.Y., Hendrycks, D., Hamm, J., Mao, Z.M.: Certified adversarial defenses meet out-of-distribution corruptions: Benchmarking robustness and simple baselines. arXiv preprint arXiv:2112.00659 (2021)

[31] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)

[32] Wen, W., Yan, F., Chen, Y., Li, H.: Autogrow: Automatic layer growing in deep convolutional networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 833–841 (2020)

[33] Wu, L., Liu, B., Stone, P., Liu, Q.: Firefly neural architecture descent: a general approach for growing neural networks. Advances in neural information processing systems **33**, 22373–22383 (2020)

[34] Wu, L., Wang, D., Liu, Q.: Splitting steepest descent for growing neural architectures. Advances in neural information processing systems **32** (2019)

[35] Yuan, X., Savarese, P., Maire, M.: Growing efficient deep networks by structured continuous sparsification. arXiv preprint arXiv:2007.15353 (2020)

[36] Yuan, X., Savarese, P.H.P., Maire, M.: Accelerated training via incrementally growing neural networks using variance transfer and learning rate adaptation. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=H1a7bVVnPK

[37] Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8697–8710 (2018)

Acknowledgements

This work was supported by the Center for the Co-Design of Cognitive Systems (CoCoSys) funded by the Semiconductor Research Corporation (SRC) and the Defense Advanced Research Projects Agency (DARPA).

Appendix / Supplemental material

A Training Setup

Datasets and Architectures: All results are shown on CIFAR-10 and CIFAR-100 [18] ($\mathcal{D}_{\text{in}}$) datasets. CIFAR-10-C and CIFAR-100-C [12] ($\mathcal{D}_{\text{out}}$) are used to benchmark corruption robustness. CNN architectures MobileNet-V1[15], VGG-19[29], ResNet-18[10] are employed to demonstrate results.

Hyperparameters: The setup for growth and robust augmentation follows closely with what is described in Firefly [33] and AugMix [13], respectively. The parametric training is done for 160 epochs using a batch-size of 128 and an initial learning rate of 0.1. The learning rate scheduler decays by 0.1 at half and three-fourths of the total number of epochs. We use the Swish loss function for MobileNet-V1 as used in [33], while employing ReLU for the other two networks. Instead of using three fully-connected layers at the end of VGG-19, we use only one as done in [33]. Stochastic Gradient Descent (SGD) optimizer is used with momentum 0.9 and weight decay 10^{-4}. As for the standard growth process, we use a Root Mean Square Propagation (RMSprop) optimizer with momentum 0.9, alpha 0.1 and initial learning rate of 9×10^{-5}. The number of workers is chosen as 4. For ERA, $(W, D, J) = (1, 3, 4)$ is picked. The augmentation transforms $\mathcal{T}$ are same as that of AugMix [13]. As specified in AugMix, we also do not use any augmentations which are directly present in the corrupted test dataset.

In case of OSG, the initial backbone f_0 is chosen as a network with $w_l = 45$ for all $l = \{1, ..., L\}$ and is thus extremely small. The number of randomly initialized neurons at each growth stage is 70. The growth ratio and epochs used at each stage is shown in Table 1. We ensure that $\mathcal{E}_2$ of GEARnn-1 and $\mathcal{E}_r$ of GEARnn-2 are same for a fair comparison.

Jetson Training: The two changes to the GEARnn algorithm when implementing on NVIDIA Jetson Xavier are - one we use $j = 3$ instead of $j = 4$, and two, we allow only 40 randomly initialized new neurons per layer in the growth step (as compared to 70 in [33]). These measures are taken to stay within the memory constraints of the Edge device. We also reduce the batch size (and learning rate) appropriately in case the above measures are insufficient.

Metrics: Clean accuracy $\mathcal{A}_{\text{cln}}(\%)$ measured on clean test data $\mathcal{D}_{\text{in}}$, and robust accuracy $\mathcal{A}_{\text{rob}}(\%)$ measured on corrupted test data $\mathcal{D}_{\text{out}}$, are used as accuracy metrics (both computed using RobustBench [3]). The number of floating-point parameters (model size), number of floating-point operations (FLOPs) per inference, wall-clock training time t_{tr} (in minutes), per-sample wall-clock inference time t_{inf} (in seconds), average power consumption P_{avg} (in Watt) and energy consumption E (in Joule) are used as the efficiency metrics. Size $(\%)$ represents the fraction of the full model size. In case of growth algorithms, training times include both the time taken for training and growth. The power is measured from the Quadro and Jetson using Nvidia-SMI [25] and Jetson Stats [1], respectively, and the energy E is computed by summing the power values polled every second.

Hardware: For the server-based experiments, we use one NVIDIA Quadro RTX 6000 GPU with 24GB RAM, 16.3 TFLOPS peak performance and an Intel Xeon Silver 4214R CPU. This machine is referred to as "Quadro". For the Edge-based experiments, we use the NVIDIA Jetson Xavier NX [24] which has a Volta GPU with 8GB RAM, 21 TOPS peak performance and a Carmel CPU. We refer to this device as "Jetson".

Baselines: In the absence of prior work on robust growth, we propose our own baselines Small ($\mathcal{D}_{\text{in}}$) and Small ($\mathcal{D}_{\text{aug}}$), both of which use 160 training epochs to be consistent with [4]. They are networks with the same size and topology as the final GEARnn-2 network (f^*_{2p} in Fig. 2) initialized randomly and trained on clean and augmented data (AugMix [13], unless specified otherwise), respectively.

We pick Small ($\mathcal{D}_{\text{aug}}$) as the main baseline for a fair comparison with GEARnn as it depicts a typical private-Edge training scenario. We do not compare with compression techniques since they have been shown to have worse training efficiency compared to growth [35], and require a robust-trained full baseline, and this is clearly more expensive than training Small ($\mathcal{D}_{\text{aug}}$) (see Fig. 1).

B Results on other Network Architectures

Results for MobileNet-V1 and ResNet-18 on Quadro are shown in Table 6.

Table 6: Comparison of accuracies, and efficiency between the baselines and GEARnn for MobileNet-V1 and ResNet-18 using CIFAR-10, CIFAR-100 and Tiny ImageNet on Quadro. See Appendix E for robustness comparison between Small ($\mathcal{D}_{\text{aug}}$) and GEARnn-2 at similar training cost.

Architecture		CIFAR-10					CIFAR-100					Tiny ImageNet				
(full model size)	Method	Size (%)	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Training t_{tr}(min)	Training E(kJ)	Size (%)	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Training t_{tr}(min)	Training E(kJ)	Size (%)	Accuracy $\mathcal{A}_{\text{cln}}(\%)$	Accuracy $\mathcal{A}_{\text{rob}}(\%)$	Training t_{tr}(min)	Training E(kJ)
MobileNetV1 (3M)	Small ($\mathcal{D}_{\text{in}}$)	8	92.28	66.31↓	42	192	8	67.66	39.04↓	45	274	8	55.13	18.48↓	262	2030
	Full ($\mathcal{D}_{\text{in}}$)	100	92.72	70.12↓	31	145	100	68.31	43.39↓	32	158	100	57.55	20.97↓	175	1450
	Small ($\mathcal{D}_{\text{aug}}$)	8	**92.90**	83.21	211	1130	**8**	**68.88**	**54.95**	212	1330	**8**	**56.46**	28.17	765	7200
	Full ($\mathcal{D}_{\text{aug}}$)	100	92.76	**85.61**	232	1070	100	65.41	53.89	228	1070	100	55.20	**31.54**	565	4330
	GEARnn-1	**7**	90.64	80.71	88	379	**8**	65.07	51.46	93	651	**8**	54.57	27.46	506	4410
	GEARnn-2	8	91.35	81.96	**56**	**270**	**8**	67.95	53.28	**72**	**432**	**8**	56.16	28.56	**429**	**3565**
ResNet-18 (12M)	Small ($\mathcal{D}_{\text{in}}$)	6	93.34	68.85↓	61	546	7	68.74	40.83↓	67	490	7	54.72	18.11↓	381	3390
	Full ($\mathcal{D}_{\text{in}}$)	100	94.46	72.74↓	86	818	100	74.75	47.15↓	77	761	100	62.52	22.95↓	484	6010
	Small ($\mathcal{D}_{\text{aug}}$)	**6**	94.18	86.50	217	1730	**7**	71.97	57.30	219	1250	**7**	54.5	25.74	1103	12400
	Full ($\mathcal{D}_{\text{aug}}$)	100	**94.76**	**88.25**	223	2140	100	**73.70**	**61.74**	219	2220	100	**60.76**	**33.67**	1429	17100
	GEARnn-1	**6**	92.36	83.86	108	747	8	69.15	55.62	142	1020	**7**	53.17	24.93	898	9100
	GEARnn-2	**6**	93.14	84.45	**77**	**567**	**7**	70.94	56.54	**97**	**905**	**7**	54.79	26.64	**649**	**7270**

C Ablation study

Impact of different GEARnn components are highlighted in Fig. 7 to show the efficacy of the choice. The training loss curves in Fig. 8 indicate why GEARnn-2 achieves good robustness efficiently.

Figure 7: Impact of GEARnn components for CIFAR-100 using VGG-19 on Quadro.

Phase-1		Phase-2		$\mathcal{A}_{\text{rob}}(\%)$	t_{tr}(min)	E (kJ)
vanilla	OSG	AugMix	ERA			
✓				38.72	18	161
	✓			38.01	16	118
		✓		46.50	62	385
			✓	46.13	46	406
✓		✓		53.74	79	534
	✓		✓	**54.31**	64	515

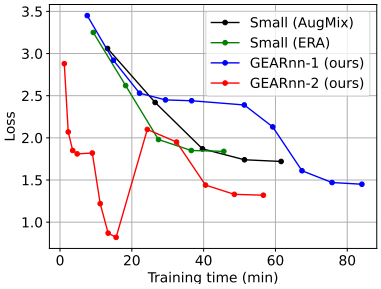


Figure 8: Comparison of losses for 2-Phase (GEARnn-2) and 1-Phase (rest) approaches. It highlights that GEARnn-2 loss converges to the minimum faster than the other approaches.

D Diagnostics of Robust Augmentation Methods

In this section, we investigate which aspects of the robust augmentation framework described in Section 4.2 contribute most to the robustness while being training efficient. Table 7 shows different modifications of the stochastic chains obtained by varying (W, D, J) values. It can be observed that the basic version with $(W, D, J) = (1, 1, 0)$ (uses only standard cross entropy loss with the label and augmented data as input) has the least training time, but suffers a significant drop in $\mathcal{A}_{\text{rob}}$ compared to standard AugMix. Crucially, we note that increase in D and J has more impact on robustness at a lesser training cost compared to W. For ERA, we pick the modification with $(W, D, J) = (1, 3, 4)$ as it provides the highest robustness while simultaneously reducing training time over AugMix.

Experiment	W	D	J	$\mathcal{A}_{\text{rob}}(\%)$	t_{tr}(min)
Basic	1	1	0	77.74	**10**
+ width	3	1	0	78.51	16
+ depth	1	3	0	80.31	12
+ JSD-3	1	1	3	82.43	20
+ width + depth	3	3	0	80.47	21
+ width + JSD-3	3	1	3	82.27	32
+ depth + JSD-3	1	3	3	83.67	22
+ depth + JSD-2	1	3	2	82.41	13
+ depth + JSD-4	1	3	4	**84.10**	29
AugMix [13]	3	3	3	84.05	41

Table 7: Impact of training AugMix-variants on the robust accuracy and training time. Network f_2 from OSG is used as the starting network and $\mathcal{E}_r$ = 40. All the methods are implemented for CIFAR-10 and 5% VGG-19 network on Quadro. W, D, J represent the width, depth and consistency samples used in the stochastic chains.

E Impact of Robust Training Epochs

In Section 6 and Fig. 6 we observed that GEARnn-2 can achieve high robustness even when the robust training epochs are low. This is due to better initialization provided by OSG. We show the same results ablated for both VGG-19 and MobileNet-V1 for CIFAR-10 and CIFAR-100 in Fig. 9.

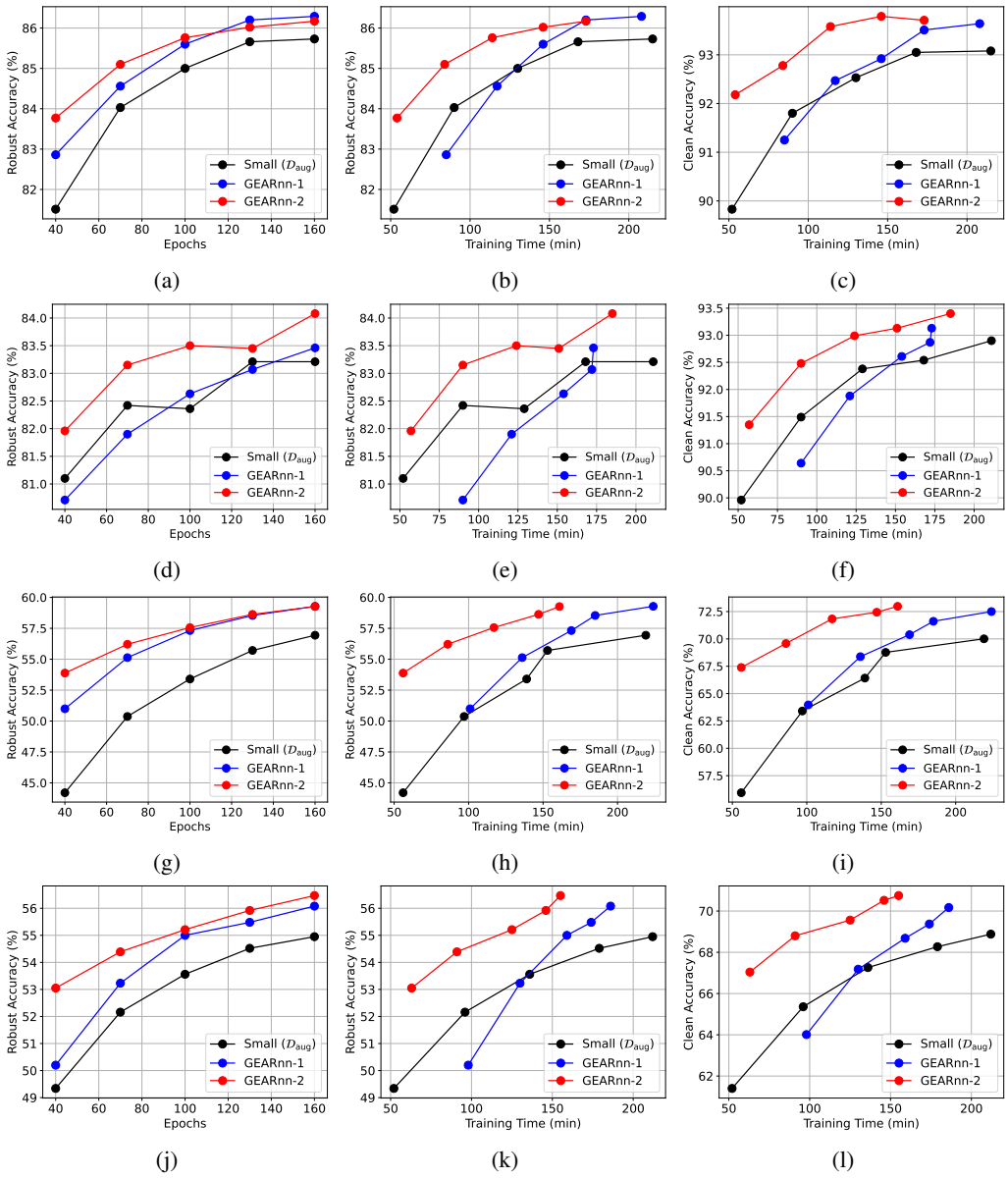


Figure 9: Plots (a)-(c) are implemented for VGG-19/CIFAR-10, (d)-(f) are for MobileNet-V1/CIFAR-10, (g)-(i) are for VGG-19/CIFAR-100, and (j)-(l) are for MobileNet-V1/CIFAR-100 on Quadro. First two plots of each row indicates the robust accuracy as a function of epochs and training time respectively. The last plot in each row shows the clean accuracy as a function of training time. GEARnn-2 clearly achieves the best clean and robust accuracy at the same training cost.

F GEARnn Algorithms

Algorithm 1 GEARnn-1

Input: clean training data $\mathcal{D}_{\text{in}}$, initial backbone network f_0, growth ratio γ, set of augmentation transforms $\mathcal{T}$, training epochs $\{\mathcal{E}_1, \mathcal{E}_g, \mathcal{E}_2\}$
Output: compact and robust model f^*_{1p}
/* Phase-1: OSG */
for $e = 1, ..., \mathcal{E}_1$ **do**
    $\mathcal{D}_{\text{aug}} := \mathcal{R}(\mathcal{D}_{\text{in}}|\mathcal{T})$ // ERA
    $f_1 \leftarrow \arg\min_f \mathcal{L}_{\text{aug}}(f, \mathcal{D}_{\text{aug}}|f_0)$ // backbone robust training
end for
$f_g \leftarrow \mathcal{G}(f_1|\gamma, \mathcal{D}_{\text{aug}}, \mathcal{L}_{\text{aug}}, \mathcal{E}_g)$ // augmented growth
for $e = 1, ..., \mathcal{E}_2$ **do**
    $\mathcal{D}_{\text{aug}} := \mathcal{R}(\mathcal{D}_{\text{in}}|\mathcal{T})$ // ERA
    $f_2 \leftarrow \arg\min_f \mathcal{L}_{\text{aug}}(f, \mathcal{D}_{\text{aug}}|f_g)$ // grown-network robust training
end for
$f^*_{1p} \leftarrow f_2$
return f^*_{1p}

Algorithm 2 GEARnn-2

Input: clean training data $\mathcal{D}_{\text{in}}$, initial backbone network f_0, growth ratio γ, set of augmentation transforms $\mathcal{T}$, training epochs $\{\mathcal{E}_1, \mathcal{E}_g, \mathcal{E}_2, \mathcal{E}_r\}$
Output: compact and robust model f^*_{2p}
/* Phase-1: OSG */
for $e = 1, ..., \mathcal{E}_1$ **do**
    $f_1 \leftarrow \arg\min_f \mathcal{L}_{\text{CE}}(f, \mathcal{D}_{\text{in}}|f_0)$ // backbone clean training
end for
$f_g \leftarrow \mathcal{G}(f_1|\gamma, \mathcal{D}_{\text{in}}, \mathcal{L}_{\text{CE}}, \mathcal{E}_g)$ // clean growth
for $e = 1, ..., \mathcal{E}_2$ **do**
    $f_2 \leftarrow \arg\min_f \mathcal{L}_{\text{CE}}(f, \mathcal{D}_{\text{in}}|f_g)$ // grown-network clean training
end for
/* Phase-2: Train */
for $e = 1, ..., \mathcal{E}_r$ **do**
    $\mathcal{D}_{\text{aug}} := \mathcal{R}(\mathcal{D}_{\text{in}}|\mathcal{T})$ // ERA
    $f^*_{2p} \leftarrow \arg\min_f \mathcal{L}_{\text{aug}}(f, \mathcal{D}_{\text{aug}}|f_2)$ // grown-network robust training
end for
return f^*_{2p}