Reinforced Compressive Neural Architecture Search for Versatile Adversarial Robustness

Anonymous Author(s)

ABSTRACT

Prior neural architecture search (NAS) for adversarial robustness works have discovered that a lightweight and adversarially robust neural network architecture could exist in a non-robust large teacher network, generally disclosed by heuristic rules through statistical analysis and neural architecture search. However, heuristic methods cannot uniformly handle different adversarial attacks and "teacher" network capacity. To solve this challenge, we propose a Reinforced Compressive Neural Architecture Search (RC-NAS) for Versatile Adversarial Robustness. Specifically, we define task settings that compose datasets, adversarial attacks, and teacher network information. Given diverse tasks, we conduct a novel dual-level training paradigm to push the model to achieve the best robust accuracy for the target task setting, and also retain the potential to fit into other versatile tasks easily. Experiments show that our framework could achieve adaptive compression towards different initial teacher networks, datasets, and adversarial attacks, resulting in a more lightweight and adversarially robust architecture compared to baselines adopting heuristic rules. We also provide a theoretical analysis to explain why the RL-guided adversarial architectural search helps adversarial robustness over standard adversarial training methods.

KEYWORDS

Adversarial Robustness, Neural Architecture Search, Compression

ACM Reference Format:
Anonymous Author(s). 2024. Reinforced Compressive Neural Architecture Search for Versatile Adversarial Robustness. In . ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

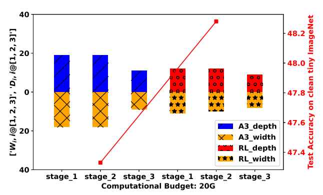
Deep neural networks (DNNs) have benefited many real-world applications, such as image classification [16], object detection [18], and natural language processing [23]. However, standard DNNs are vulnerable to adversarial attacks, raising an effective remedy to include deeper and/or wider blocks along with adversarial training [19, 22, 29, 31, 33]. Since such strategies may incur significant computational overhead, recent efforts have been devoted to locating lightweight architectures that are robust to different adversarial attacks through neural architecture search (NAS) [6, 9, 12, 13, 24, 35].

One mainstream direction in NAS is to leverage a generative process to seek the best-performed network architectures based on a manually designed library of architectural ingredients. However, given the complex nature of DNN architectures coupled with the diverse types of adversarial attacks, such a process can become too costly to cover various aspects, making it hard to guarantee a good adversarially robust performance. Another line of work suggests that there exists an optimal architectural configuration for adversarial robustness in a large non-robust "teacher" architecture, which enjoys a smaller parameter size and better robustness [12, 24]. Consequently, network-to-network (N2N) compression could be conducted to achieve adversarial robustness and has shown promising progress in recent works [12, 13]. For example, Huang et al. [13] investigated over 1,000 network architectures randomly sampled from some large teacher networks and selected top-ranked robust sub-networks. Empirically, they derived a set of useful rules that can help to guide the design of robust ResNet (`RobustResNet`) architectures from different teacher networks and computation budgets, varying from 5G to 40G. However, most of these rules are derived in a heuristic way, offering no guarantee of achieving an optimal trade-off between compression ratio and adversarial accuracy as the learning environment changes, which thus may lead to suboptimal performance. As shown in Figure 1, `RobustResNet` [13] suffers from a lower adversarial accuracy due to adopting the fixed configuration rules across different attacks and computation budgets.

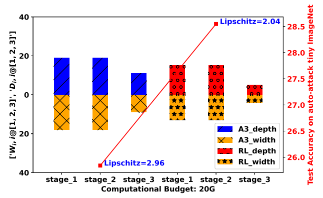
To overcome the key limitations of existing solutions, we propose a novel reinforcement learning (RL) framework, referred to as Reinforced Compressive Neural Architecture Search (`RC-NAS`), that leverages the flexibility of a specially designed RL agent supported by a powerful dual-level training paradigm to perform a systematic search over a rich and complex space of architecture configurations. The trained RL agent can quickly adapt to the highly diverse attack scenarios and locate a compressed student sub-network with guaranteed adversarial robustness while meeting the computational budget constraints. The ability to automatically adjust to distinct attack scenarios and adaptively compress the teacher network in different ways (instead of following fixed rules) is a critical step towards realizing truly versatile adversarial robustness that significantly advances the state of the art.

To achieve versatile adversarial robustness, it is essential to identify the key characteristics from different attack scenarios and train the RL agent to recognize them to perform adaptive N2N compression given a specific attack setting. To this end, the proposed RL framework collectively considers important features of the teacher network, the nature of the dataset, the level of adversarial attack, and the overall computational budget. Based on different attack conditions, the compression ratio of each stage should be adaptive, and correspondingly, the model should take different compression operations for each stage to achieve the best compression and robust accuracy trade-off. We further leverage the Lipschitz regularity to capture the level of adversarial attack being applied to a dataset.

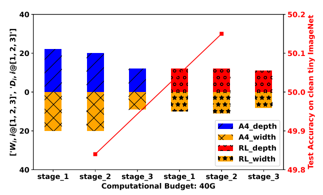
Conference'24, August 2024, Barcelona, Spain
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn



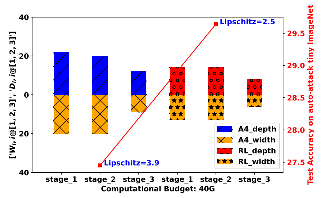
(a) 20G on clean



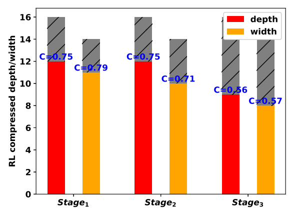
(b) 20G on auto-attack



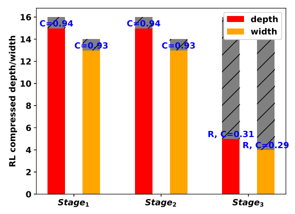
(c) 40G on clean



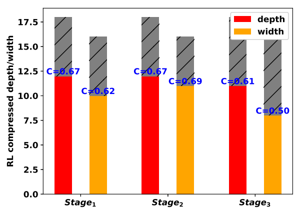
(d) 40G on auto-attack



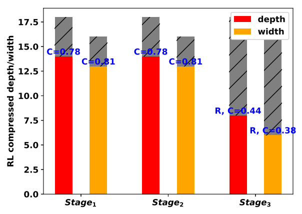
(e) WRN-46-14 on clean



(f) WRN-46-14 on auto-attack



(g) WRN-70-16 on clean



(h) WRN-70-16 on auto-attack

Figure 1: Architecture topology analysis of `RobustResNet` in different attack scenarios from the mildest (*i.e.,* clean) to the most severe (*i.e.,* auto-attack) on Tiny-ImageNet. WRN-46-14 and WRN-70-16 are leveraged as teacher networks with 20 and 40 GFLOPs computation budgets, respectively. The corresponding student networks are referred to as `RobustResNet-A3` and `RobustResNet-A4`. The entire teacher network architecture is partitioned into multiple (*e.g.,* 3) stages as in [13] and we visualize both depths and widths of the corresponding `RobustResNet` for each stage. In (a)-(d), the left three bar plots (in blue and orange) show the depths and widths in the three stages of `RobustResNet` A3 and A4 that follow the same configuration rules; the right three bar plots (in red and orange) show the adaptive configuration obtained by the proposed reinforced learning (RL) based architecture search. In (e)-(h), the grey bars denote the capacity of the corresponding teacher networks and C denotes the remaining percentage of each stage after compression.

The Lipschitz parameter is further integrated with the network topology and the overall computational budget to form a state in the simulated RL environment. Given a state, the RL agent performs an action to shrink the network by compressing the width and depth at the stage level as well as determining specific configurations (*e.g.,* convolution type, activation, and normalization) at the block level. A specially designed award function that integrates adversarial accuracy with computational budget constraint serves as the feedback to encourage the RL to seek for architectures with an optimal robust accuracy and compression trade-off.

We introduce a novel dual-level training paradigm that consists of a meta-training and a fine-tuning phases to effectively expose the RL agent to diverse attack scenarios so that it can adapt quickly to locate a sub-network customized towards the unknown attacks during the test phase. In the meta-training phase, a pool of diverse attack tasks with distinct characteristics is formulated. The RL agent is iteratively trained by sampling different tasks from the pool so that it can gain the capability to recognize key patterns from different combinations of these key factors and perform adaptive N2N compression. In the second phase, the agent is given some specific attack scenario and it performs fine-tuning by leveraging the meta-trained model as the starting point to achieve quick and effective adaptation. Figure 1 (a)-(d) shows that the proposed RL framework can adapt to different attack scenarios by finding highly customized robust sub-networks, instead of relying on a fixed set of configuration rules as in existing works. As a result, it leads to much improved adversarial accuracy with a better compression ratio. Furthermore, as shown in (e)-(h), highly distinct compression ratios are applied to different stages of the teacher network and an overall higher compression ratio is applied to a larger teacher network. More interestingly, the compression ratio also vary dramatically based on the level of attacks: for a more severe attack (*i.e.,* auto-attack), the first two stages are compressed much less (to capture the subtle changes in the input) while the last stage is compressed more significantly (to reduce the perturbation caused by the attack). Our theoretical analysis provides further insights on the compression behavior.

To the best of our knowledge, this is the first effort to provide a principled RL framework for searching an adversarially robust architecture in a non-robust large teacher network (network-to-network compression for adversarial robustness). Our main contributions are summarized below:

- a novel Reinforced Compressive Neural Architecture Search (`RC-NAS`) framework that leverages the flexibility of reinforcement learning to explore a rich and complex space of architectures for lightweight and adversarially robust sub-networks,
- a simulated RL environment equipped with a specially designed state encoder and an award function, allowing the RL agent to encode key ingredients from the teacher architecture, the dataset, level of adversarial attack, and the computational budget,
- dual-level RL training to enable quick adaption to specific attack settings by exposing the RL agent to diverse attack scenarios,
- deeper theoretical analysis that reveals why the RL driven N2N compression can lead to improved adversarial robustness.

We conduct extensive experiments to demonstrate the effectiveness of the proposed `RC-NAS` framework over different input conditions, including different teacher network architecture, with pairing computation budgets (5G–40G), and visual learning tasks of varying difficulty (CIFAR-10, CIFAR-100, and Tiny-ImageNet). For the adversarially trained RL compressed network, we compare it with the latest N2N compression baselines for adversarial robustness and show its superior performance across different teacher network capacities, datasets, and adversarial attack test conditions. We also emphasize that such good performance is attributed to the novel RL framework and its unique dual-level training paradigm for learning from diverse input conditions and taking adaptive compression actions adaptively, which is verified by multiple ablative studies and statistical analysis of RL selected robust sub-network architectures.

2 RELATED WORK

Neural Architecture Search (NAS). There has been much work on exploring the rich design space of neural network architectures [6, 14, 35, 36]. The principal aim of previous work in architecture search has been to build models that maximize performance on a specific dataset. There has been increasing interest in conduct a N2N style architecture search to achieve adversarial robustness [9, 12, 13]. For example, Wu et al. theoretically analyze why wider networks tend to have worse perturbation stability on linearized neural tangent kernels and develop a width adjusted regularization (WAR) algorithm to address that [30]. Huang et al. emphasize that a higher model capacity does not necessarily improve adversarial robustness, especially in the last stage and there exists an optimal architectural configuration for adversarial robustness under the same parameter budget [12]. To this end, residual networks have been intensively analyzed by considering architecture design at both the block level and the network scaling level [13]. A robust residual block and a compounding scaling rule have been derived to properly distribute depth and width at the desired computation budget. However, those proposed heuristic rules for N2N compression are either too general or too specific for diverse adversarial learning scenarios. A flexible way to achieve a robust architecture that can adapt to the unique characteristics of each attack scenario is in demand to achieve truly versatile adversarial robustness, which is the focus of our work.

Network pruning. Pruning-based methods preserve the weights that matter most and remove the redundant weights [8, 10, 17]. While pruning-based approaches typically operate on the weights of the model, our approach operates on a much larger search space over both model weights and model architecture. Recently some works have combined adversarial learning with network pruning such as Hydra [24] and ADMM [32], but none of them have paid attention to architecture search. Instead, our work focus on reinforced neural architecture search for adversarial robustness, thus providing more flexible architectural design choices and enjoying a compressed parameter space at the same time.

Reinforcement learning. Deep Reinforcement Learning has been extensively applied in game agent training [20], natural language contextual understanding [25], causal relationship inference [34], image classification [11], object detection [3], time series analysis [27] and information retrieval [28]. In the NAS domain, there are some works [2, 36] focusing on designing an RL agent to sample a sub-network within a pre-defined architecture search space. For example, Ashok et al. perform student-teacher knowledge distillation for clean accuracy on small datasets [2]. However, none of existing efforts pay attention to the relationship between the adversarial robustness and the reinforced network-to-network compression, which is main design focus of our RL framework.

3 METHODOLOGY

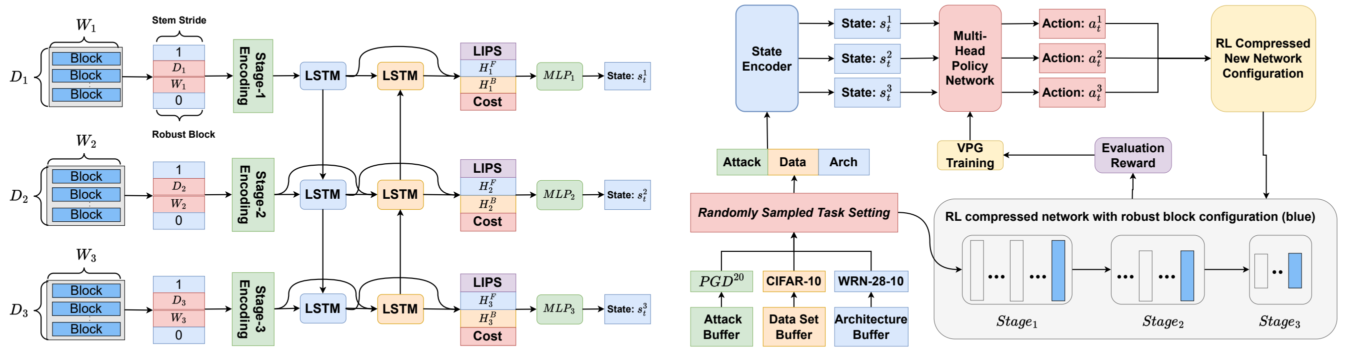
Overview. The overall goal of the proposed RC-NAS framework is to train a RL agent so that it can perform adaptive N2N compression of a large teacher network to achieve lightweight sub-networks robust to specific adversarial attacks. Given a different attack scenario, the RL agent needs to recognize its key characteristics and formulates a customized policy to generated compression actions. To this end, we propose a dual-level training paradigm and employ a meta-training phase to expose the RL agent to diverse attack scenarios. For the action design, we want our RL agent to control both macro stage-level width/depth configuration and micro block-level details, such as convolution types, activation and normalization. To support RL training, we define a formal Markov Decision Process that provides the key building blocks of the RL environment.

3.1 Markov Decision Process

The Markov Decision Process (MDP) for our proposed RC-NAS defines its own state, action, reward and state transition function:

- **State**: State s_t is an embedding encoding the input teacher network topology, adversarial attack level, dataset complexity, and corresponding computation costs.
- **Action**: Action a_t is an embedding designating RL compressed stage-level and block-level configurations given the input architecture for the current time step.
- **Reward**: Reward r_t is the trained RL-compressed sub-network's evaluation accuracy on a separate evaluation set of the same difficulty as input training set, weighted by compression ratios and normalized by the initial teacher network's performance. Also we consider the desired computation budget and set an annealing penalty if it is not satisfied.
- **State Transition Function:** Transition function $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$ is achieved by multiple buffers, i.e. an *architecture* buffer which stores all the candidate teacher architectures to compress, a *Data* buffer containing small/medium/large dataset choices, and an *Attack* buffer consisting of different adversarial attack methods. For each time step, we randomly sample one architecture-dataset-attack combination from these buffers as the new input task setting for state encoding.

In each time step, we sample the input teacher architecture, adversarial attack, as well as a dataset and encode them into the next state s_{t+1} by leveraging the state encoder. Then, the multi-head policy network takes s_{t+1} to generate a compression action a_{t+1} that specifies the compression operations, such as the remaining percentage of depth and width for each stage, as well as the application of robust block configurations. After getting the newly sampled sub-network, it is included into the *architecture* buffer and the RL agent moves to the next step. For RL training, we train the RL sampled sub-network on the training set with standard adversarial training and test its adversarial performance on corresponding



(a) State Encoder **(b) RL training framework**

Figure 2: Overview of the RC-NAS framework

evaluation set of the same input task setting. The reward is calculated based on the adversarial accuracy normalized by the robust accuracy from the teacher network, and weighted by the compression ratio comparing to the teacher network. We also consider an annealing penalization if the computation budget is not satisfied. The whole RL framework is trained by vanilla policy gradient based on the above defined reward. We introduce the detailed RL settings in the following section.

3.2 Reinforced Neural Architecture

Attack task formulation An attack task is formed by combining an adversarial attack method $\mathcal{P}$, a dataset D, and an initial teacher architecture f_ψ sampled their corresponding buffers. An evaluation set D_{eval} is further separated from the dataset for reward evaluation (we assign 256 data samples to the evaluation set in our experiments). After the RL agent is fully trained, it will sample a sub-network from the target task's teacher network f_ψ, and then the selected sub-network will be trained using a standard adversarial training process on the clean training set D_{train} and evaluated on $\mathcal{P}$-attacked test set D_{test} from the target task.

State encoder. As shown in Figure 2a, the state encoder takes the initial teacher network topology as input, including each stage's width (scalar), depth (scalar), stride step of its first layer indicating whether to down-sample the input feature map into half from the last stage output (binary), and the application of robust block [13] inside each stage (binary). We split above information into stage encodings which summarize each stage's related information. These stage encodings $\mathtt{ST}_i, \forall i \in [1, N_{stage}]$ will go through a bi-directional LSTM to output forward and backward encodings H_i^F and H_i^B, respectively. The forward encoding summarizes all previous stage information while the backward encoding considers the afterward stage information beyond the current stage.

To capture the level of adversarial attack, we propose to evaluate the Lipschitz coefficient [26] of the attacked dataset with respect to the teacher network. This can be achieved through three steps: 1) Train the teacher network using a standard adversarial training process on the training set. 2) Perform adversarial attack on the clean trained teacher network with evaluation set data applying the adversarial attack method from the same input task setting. 3) Calculate the Lipschitz coefficient of each adversarial attacked data instance on the evaluation set by comparing to the network output from the same clean instance,

$$\mathtt{LIPS} = \frac{\|f_\psi(\mathcal{P}(x_i;\hat{\epsilon})) - f_\psi(x_i)\|_1}{\|\mathcal{P}(x_i;\hat{\epsilon}) - x_i\|_\infty}, x_i \in D_{eval} \tag{1}$$

where f_ψ is the adversarially trained teacher network, $\mathcal{P}$ is the chosen adversarial attack, and D_{eval} is the evaluation set.

Finally, the capacity of the current input teacher network is represented by its inference speed cost on every data instance of the evaluation set, which is measured by GFLOPs, resulting into another embedding vector $\mathtt{CT}$. By concatenating all four embedding vectors, we generate the state embedding $\mathtt{concat}(\mathtt{LIPS}, H_i^F, H_i^B, \mathtt{CT})$. Through a multi-layer perception module $\mathtt{MLP}_i$ we transform the a state embedding into the RL state s_t^i,

$$s_t^i = \mathtt{MLP}_i\left(\mathtt{concat}(\mathtt{LIPS}, H_i^F, H_i^B, \mathtt{CT})\right) \tag{2}$$

Multi-head policy network. Multi-head policy network $f_\phi(\cdot)$ functions as the RL actor to generate actions given state $s_t = \mathtt{concat}\{s_t^i, i \in [1, N_{stage}]\}$. The generated action $a_t = \mathtt{concat}\{a_t^i, i \in [1, N_{stage}]\}$ where each a_t^i corresponding to $stage_i$ is a four dimensional vector and designates four different compression operations for each stage. The policy network is multi-head after a shared feature extraction module $\mathcal{E}$. Specifically, the our heads are to generate the two pairs of means: $\mu_i = (\mu_{i,1}, \mu_{i,2})^\top$ and a covariance matrix $\Sigma_i = \begin{bmatrix} \sigma_{i,1}^2 & 0 \\ 0 & \sigma_{i,2}^2 \end{bmatrix}$ for constructing a two-dimensional Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$, to sample the remaining percentage of the width and depth after RL compression. We use a diagonal co-variance matrix Σ_i by assuming that width and depth are independent to each other. Another two heads are used to generate the probabilities $p = (p_{i,1}, p_{i,2})$ for sampling two binary signals through a multi-Bernoulli distribution Ber to designate whether to half down-sample the input feature map in the stage's first layer through the first binary signal and whether to apply the robust block configuration in that stage through the other binary signal. For those Gaussian and Bernoulli heads with parameters $\phi_\mathcal{N}$ and $\phi_\mathcal{B}$, we train them with vanilla policy gradient using the reparametrization trick [15], and the gradients from two heads will both trace back to and update the shared feature extraction module parameters $\phi_{\mathcal{E}}$. We formulate this process below:

$$
\begin{aligned}
&(\mu_i, \Sigma_i) = f_{\phi_{\mathcal{E}}, \phi_{\mathcal{N}}}(s_t^i), \quad p_i = f_{\phi_{\mathcal{E}}, \phi_{\mathcal{B}}}(s_t^i) \\
&a_t^i = \mathtt{concat}(\sigma(\alpha_i), \beta_i), \alpha_i \sim \mathcal{N}(\cdot|\mu^i, \Sigma^i), \beta_i \sim Ber(\cdot|p_i) \qquad (3)
\end{aligned}
$$

where σ is the sigmoid activation function.

RL reward design. The evaluation reward is defined in Eq. (4). First, the generated network compressed by the RL actions is trained on the training set of the given task using standard adversarial training, and then evaluated on the adversarially attacked evaluation set of the task to get the adversarial accuracy $\tilde{A}_{RL}$. Then, the accuracy is normalized by the adversarially trained teacher network's evaluation accuracy $\tilde{A}_{teacher}$ and then weighted by a compression ratio C comparing to the teacher network, we use the quadratic form $C(2-C)$ to smoothly encourage a higher compression due to the characteristics of the quadratic curve, where $C = 1$ reaches the peak of curve $C(2-C)$. We also consider the desired computation budget CB which is measured by GFLOPs. If the average inference speed (GFLOPs) of the RL compressed sub-network on evaluation set $\zeta_{RL} = Average(Cost)$ surpasses this computation budget, the reward will be penalized by an annealing factor ϵ which starts 1 and gradually reduce to 0 as training goes. Therefore in the beginning of training, RL agent is encouraged to find the best adversarial robust architectures suitable to the task setting while not paying too much attention to the computation budget limit, and as it is trained to grasp the necessary knowledge about architecture choices, it will try to fulfill the computation budget requirements while following its summarized rules for compressing the network.

$$
r(s_t, a_t) = \begin{cases} C(2-C) \cdot \dfrac{\tilde{A}_{RL}}{\tilde{A}_{teacher}} & \zeta_{RL} \leq CB \\ \epsilon \left(C(2-C) \cdot \dfrac{\tilde{A}_{RL}}{\tilde{A}_{teacher}} + 1 \right) - 1 & otherwise \end{cases} \qquad (4)
$$

Optimization. The state encoder is pre-trained separately as Figure 5 shows in Appendix C. After its pre-training, the state encoder's weight will be frozen during RL framework training to guarantee the training stability and success. For RL framework optimization, only the actor will be updated by an RL training algorithm, namely vanilla policy gradient (VPG), through multiplying the reward with the probability of action leading to that reward. The training objective optimization of VPG is defined as Eq. (5), and the parameter ϕ of the policy network $f_\phi(\cdot)$ will be updated.

$$
\nabla_{\phi_{\mathcal{B}}, \phi_{\mathcal{N}}, \phi_{\mathcal{E}}} J = \sum_{t=1}^{T} \left[r(s_t, a_t) \times \nabla_{\phi_{\mathcal{B}}, \phi_{\mathcal{N}}, \phi_{\mathcal{E}}} (\mathcal{N}(\alpha_i|\mu_i, \Sigma_i) + Ber(\beta_i|p_i)) \right]
$$

where $\alpha_i \sim \mathcal{N}(\cdot|\mu^i, \Sigma^i), \beta_i \sim Ber(\cdot|p_i)$ (5)

3.3 Dual-level Training Paradigm

We develop a dual-level training paradigm for the RL framework training. The meta-training phase aims to expose the RL agent to diverse adversarial attack scenarios so that it can it can gain a general knowledge from learning a wide range of tasks of varying levels of attack, dataset difficulty, input teacher architecture, and computational budget. In the second phase, the agent performs fine-tuning on the target task by leveraging the meta-trained model as the starting point to achieve quick and effective adaptation.

- *Meta RL training*: In each RL time step as shown in Figure 2b, first we randomly sample a different task $task = (data, adv, teacher)$ from three initialized data, adversarial attack and architecture buffers. Then, we calculate the Lipschitz coefficient and inference cost encodings (`LIPS`, `CT`) and teacher network stage encoding $\mathtt{ST}_i, i \in [1, N_{stage}]$. We concatenate these embeddings and pass to the state encoder and get the state s_t. Through the policy network $f_\phi(\cdot)$, we get the action a_t for different stages and then conducting RL compression. For the newly RL compressed sub-network, we evaluate its reward using Eq. (4) and train the RL using Eq. (5) with VPG. Then, the sub-network will be added into the architecture buffer and move to the next time step by sampling next input task setting. When the maximum time step T is reached, we start a new RL iteration by re-initializing all the buffers until maximum RL iteration M is reached.
- *Downstream RL fine-tuning*: After meta RL training, we let the RL agent quickly adapt to the target domain by conducting a few iterations of fine-tuning. We conduct similar operations as in meta RL training and the only exception is that the target task setting could be repeatedly provided as fine-tuning task input instead of randomly sampling. We set $T = 1, M = \tilde{M}$ for an one time-step, limited iteration RL fine-tuning.

The psueduo code of the detailed training process is summarized by Algorithms 1 & 2 in the Appendix.

3.4 Theoretical Analysis

In this section, we theoretically demonstrate how the proposed RL-guided compressed student sub-network trained on adversarial samples results in a better adversarial accuracy compared to the dense network trained using standard adversarial training. To prove this, we leverage the idea of the dense mixture accumulation concept and extend the recently developed theoretical framework that justifies the effectiveness of the adversarial training to improve the robustness performance [1]. For this, first, we define the problem setup that involves the key concepts used in our theoretical analysis along with some assumptions to make the proof easier. Next, we present a lemma, demonstrating how the proposed RL-compressed student sub-network leads to the tighter gradient update bound compared to one without compression. Finally, based on the lemma, we present the main theorem that shows how the proposed RL-compressed student sub-network achieves better robust accuracy on adversarial training by ensuring the more reduction in the dense mixture components compared to the the one without compression. For brevity, we denote RL-compressed student sub-network trained using adversarial samples as S_{RL-C}^A and the one without compression as S_U^A.

Problem setup. Let us assume $\mathbf{x}_i \in \mathcal{R}^D$ indicates the i^{th} data sample that is used to train the student sub-network obtained using our proposed `RC-NAS` framework. Also, let us assume that the data sample $\mathbf{x}_i \in \mathcal{R}^D$ is generated from the sparse coding mechanism with $\mathbf{x} = \mathbf{Mz} + \hat{\epsilon}$, where $\mathbf{M} \in \mathcal{R}^{D \times D}$ is a sparse matrix whose basis functions is learned by student sub-network. Further $\mathbf{z} \in \mathcal{R}^D$ is the sparse hidden vector with sparsity defined by parameter k. $\hat{\epsilon} \in \mathcal{R}^D$ is the Gaussian noise with zero mean and standard deviation of σ_x. Also, for the sake of simplicity let us assume that the final student sub-network obtained using `RC-NAS` is a simple, two-layer symmetric neural network with ReLU activation. Then with $\Theta_{i,t}$ being the hidden weight for the i^{th} neuron at time t, the student sub-network can be represented as:

$$f_t(\Theta; \mathbf{x}, \rho) = \sum_{i=1}^{N} \left(\mathsf{ReLU}[\langle \Theta_{i,t}, \mathbf{x} \rangle + \rho_i - b_t] - \mathsf{ReLU}[-\langle \Theta_{i,t}, \mathbf{x} \rangle + \rho_i - b_t] \right) \quad (6)$$

where b_t is the bias at t^{th} at training step t, and $\rho_i \sim \mathcal{N}(0, \sigma_p^2)$ is the smoothing of the original ReLU. Then at any clean training t, the weight learned by i^{th} neuron can be decomposed as the following

$$\Theta_{i,t} \approx g_{i,t} + v_{i,t} \quad (7)$$

where $g_{i,t} = O(1)\mathbf{M}_j$ indicates pure features' contribution to produce the desired output and $v_{i,t} = \sum_{j\prime \neq j} \left[O\left(\frac{k}{D}\right) \Theta'_j \mathbf{M}'_j \right]$ indicates the dense mixture learned during the clean training in the direction of $\mathbf{M}'_j$ that are responsible to generate inaccurate response during the adversarial attack. It should be noted that we have a big portion of the robust (pure) features along with some small dense mixtures during the training process in a given neuron. Also, $j \in \mathcal{N}_j$ where $\mathcal{N}_j$ indicates the subset on which we have a highly correlated pure features. Based on this problem setup we present the following lemma for the gradient update.

Lemma 3.1. *Let T be the total iterations for the clean training and T′ be the additional iterations for the adversarial training. Considering δ be the l2-perturbation applied on input sample for the adversarial training with a radius of the perturbation defined as $||\delta||_2 \leq \tau$. Then gradient movement bound for S^A_{RL-C} is lower than that of the S^A_U. Specifically let ΔL^{RL-C}_t be the gradient movement in S^A_{RL-C} and ΔL^U_t be the gradient movement in the S^A_U, then the following inequality holds at any iteration t*

$$\Delta L^{RL-C}_t \leq \Delta L^U_t \quad (8)$$

Remark. It should be noted that in the case of the S^A_{RL-C}, we reduce the width as well as the depth of the given network. This is the same as zeroing out unnecessary edges in our dictionary matrix $\mathbf{M}$. However, zeroing out the pure (robust) features in dictionary $\mathbf{M}$ will lead to a smaller reward. Intuitively, we can infer that to maximize the reward, the RL-agent is forced to compress less important (mostly the dense mixture) components in the given network. Based on this, we have the following theorem.

Theorem 3.2. *Let $v^{T+T'}_{i,RL-C}$ be the final dense mixture component for the i^{th} neuron of S^A_{RL-C} and $v^{T+T'}_{i,U}$ be the dense mixture for the i^{th} neuron of S^A_U. Then, based on the gradient update Lemma 3.1 with high probability we have the following:*

$$\max_{i \in N} ||v^{T+T'}_{i,RL-C}||_2 \leq \max_{i \in N} ||v^{T+T'}_{i,U}||_2 \quad (9)$$

Remark. This theorem indicates that our proposed RC-NAS based sparsification mechanism on a teacher network has the potential to further lower the dense mixture components compared to without sparsification. This is because, through the sparsification, our RL-agent strives to find the sparse student sub-network that can potentially improve the adversarial accuracy by zeroing out the many non-robust entries in the dictionary $\mathbf{M}$. Please refer to the Appendix for the proof.

4 EXPERIMENTS

Datasets. We have three datasets as our test beds, CIFAR-10, CIFAR-100 and Tiny-ImageNet. CIFAR-10 and CIFAR-100 are widely used benchmark datasets in computer vision for image classification tasks. CIFAR-10 consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class, while CIFAR-100 has the same number of images but in 100 classes. In Tiny-ImageNet, there are 100,000 images divided up into 200 classes. Every image in the dataset is downsized to a 64×64 colored image. For every class, there are 500 training images, 50 validating images, and 50 test images.

Adversarial attacks. Projected Gradient Descent (PGD) [19] is an iterative FGSM method by iteratively conducting FGSM until the image is misclassified or a certain number of iterations is reached. In our setting, we try PGD^{20} attack methods with an attack radius $\hat{\epsilon} = 8/255$ and with a maximum number of iterations 20. Also, we investigate a traditional Carlini & Wagner (CW^{40}) attack [4], which utilizes two separate losses: An adversarial loss to make the generated image actually adversarial, i.e., is capable of fooling image classifiers, and an image distance loss to constrain the quality of the adversarial examples so as not to make the perturbation too obvious to the naked eye. Auto-attack [5] is a prevailing comprehensive attack method which is a parameter-free, computationally affordable, and user-independent ensemble of existing attacks.

Experimental settings. Given four initial teacher networks (WRN-28-10, WRN-34-12, WRN-46-14, WRN-70-16) corresponding to different computation budgets (5G,10G,20G,40G), we apply a range of diverse adversarial attacks targeting on the trained teacher network on the clean training set with the evaluation set and test set data. In RL training and downstream RL finetuning, we leverage the training set and adversarially attacked evaluation set only. Once we get the RL compressed sub-network architectures for the target task, we train it with standard adversarial training on clean training set and test its adversarial performance using classification accuracy on the adversarially attacked test set generated above from the same task setting. For fair comparison, all baselines (architectures found by other research works given same computation budgets) will be trained with standard adversarial training on the clean training set from the same task setting and evaluated their performance on the same test split in the target task setting. The standard adversarial training (AT) method is TRADES [33] with auto-attack as the adversarial attack choice.

Baselines. We compare our proposed RC-NAS with other latest neural architecture search works for adversarial robustness, such as RobNet-large-v2 [9], AdvRush [21], RACL [7], WRN-34-R [12] and RobustResNet A1-A4 [13] corresponding to computation budgets 5G, 10G, 20G, 40G, respectively. We also include the performance of the teacher network in the beginning of each computation budget category for reference. For fair comparison, we align the network capacity of AdvRush and RACL to different computation budgets by adjusting the number of repetitions of the normal cell N and the input channels of the first normal cell C, denoted as (N@C). The additional details of those baselines are described in Appendix F.

Table 1: Baseline Comparison on CIFAR datasets

Model	#P(M)	#F(G)	CIFAR-10				CIFAR-100			
			Clean	PGD^{20}	CW^{40}	AutoAttack	Clean	PGD^{20}	CW^{40}	AutoAttack
WRN-28-10	36.5	5.20	84.62±0.06	55.90±0.21	53.15±0.33	51.66±0.29	56.30±0.28	29.91±0.40	26.22±0.23	25.26±0.06
RobNet-large-v2	33.3	5.10	84.57±0.16	52.79±0.08	48.94±0.04	47.48±0.04	55.27±0.02	29.23±0.15	24.63±0.11	23.69±0.19
AdvRush (7@96)	32.6	4.97	84.95±0.12	56.99±0.08	53.27±0.03	52.90±0.11	56.40±0.09	30.40±0.21	26.16±0.03	25.27±0.02
RACL (7@104)	32.5	4.93	83.91±0.13	55.98±0.15	53.22±0.08	51.37±0.11	56.09±0.08	30.38±0.03	26.65±0.02	25.65±0.10
RobustResNet-A1	19.2	5.11	85.46±0.25	58.74±0.12	55.72±0.04	54.42±0.08	59.34±0.09	32.70±0.14	27.76±0.09	26.75±0.14
RC-NAS	**18.8**	**4.98**	**86.32±0.08**	**60.48±0.12**	**58.34±0.25**	**57.66±0.24**	**62.33±0.19**	**34.9±0.15**	**29.95±0.27**	**29.35±0.25**
WRN-34-12	66.5	9.60	84.93±0.24	56.01±0.28	53.53±0.15	51.97±0.09	56.08±0.41	29.87±0.23	26.51±0.11	25.47±0.10
WRN-34-R	68.1	19.1	85.80±0.08	57.35±0.09	54.77±0.10	53.23±0.07	58.78±0.11	31.17±0.08	27.33±0.11	26.31±0.03
AdvRush (10@96)	67.5	18.33	85.33±0.13	57.08±0.12	54.53±0.14	52.67±0.15	57.14±0.13	30.45±0.15	26.54±0.15	26.22±0.16
RACL (10@104)	67.9	18.75	84.82±0.18	56.38±0.13	53.89±0.16	52.23±0.16	56.78±0.12	30.22±0.15	26.35±0.17	25.79±0.19
RobustResNet-A2	39.0	10.8	85.80±0.22	59.72±0.15	56.74±0.18	55.49±0.17	59.38±0.15	33.0±0.17	28.71±0.19	27.68±0.21
RC-NAS	**37.4**	**9.67**	**86.84±0.18**	**61.08±0.35**	**60.45±0.24**	**58.68±0.15**	**63.15±0.22**	**36.96±0.25**	**30.55±0.36**	**30.79±0.33**
WRN-46-14	128	18.6	85.22±0.15	56.37±0.18	54.19±0.11	52.63±0.18	56.78±0.47	30.03±0.07	27.27±0.05	26.28±0.03
AdvRush (16@100)	131	23.39	86.38±0.05	57.05±0.12	55.08±0.21	54.15±0.17	57.95±0.28	31.25±0.14	28.39±0.12	28.24±0.13
RACL (16@108)	132	24.12	85.45±0.08	56.58±0.15	54.68±0.24	53.29±0.13	57.13±0.27	30.78±0.17	27.85±0.15	27.54±0.18
RobustResNet-A3	75.9	19.9	86.79±0.09	60.10±0.14	57.29±0.25	55.84±0.15	60.16±0.22	33.59±0.19	29.58±0.12	28.48±0.19
RC-NAS	**68.5**	**18.4**	**88.46±0.15**	**62.15±0.11**	**60.88±0.09**	**59.21±0.21**	**64.75±0.18**	**37.13±0.24**	**31.79±0.25**	**31.75±0.16**
WRN-70-16	267	38.8	85.51±0.24	56.78±0.16	54.52±0.16	52.80±0.14	56.93±0.61	29.76±0.17	27.20±0.16	26.12±0.24
AdvRush (22@100)	266	41.75	86.11±0.12	56.12±0.15	54.17±0.09	53.38±0.20	56.13±0.19	30.08±0.16	27.16±0.18	27.04±0.19
RACL (22@110)	264	40.96	84.88±0.24	56.17±0.18	54.49±0.26	52.77±0.14	56.79±0.19	30.05±0.23	27.13±0.14	26.88±0.22
RobustResNet-A4	147	39.4	87.10±0.15	60.26±0.13	57.9±0.18	56.29±0.12	61.66±0.64	34.25±0.19	30.04±0.18	29.00±0.28
RC-NAS	**129**	**35.8**	**89.22±0.12**	**62.58±0.15**	**61.30±0.22**	**59.98±0.16**	**65.47±0.52**	**37.74±0.23**	**31.96±0.22**	**32.02±0.31**

4.1 Results and Comparison

For target task settings, we use CIFAR-10, CIFAR-100 and Tiny-ImageNet as dataset choices, and train all baseline models under four computation budgets using standard adversarial training on the same clean training set and test their adversarial performance on the clean test set or test set with three different adversarial attack categories (PGD^{20}, CW^{40}, AutoAttack). For the compressed sub-networks selected by the RL agents which are fine-tuned given different target tasks, we train them with standard adversarial training and report its their respective test performance on the corresponding task setting, i.e. CIFAR-10, CIFAR-100 test set with no attack or three different kinds of adversarial attacks, same as other baselines. We summarize the comparison results of CIFAR-10 and CIFAR-100 in Table 1, and results of Tiny-ImageNet in Table 2. From Table 1 and 2, our model consistently achieves better classification accuracy under different computation budgets and adversarial attacks on the test sets, either for CIFAR-10, CIFAR-100 or Tiny-ImageNet, compared to all the other baselines of similar budgets. The RL decided sub-network is not only superior in robust accuracy against adversarial attacks, but also smaller regarding its parameter size (See column $\#P(M)$) and achieves faster inference speed, denoted by GFLOPs in column $\#F(G)$.

4.2 Ablation Study

4.2.1 Effectiveness of RL guided exploration. We investigate the effectiveness of the RL guided architectural exploration by replacing it with other existing techniques, including advanced adversarial training methods (TRADES, SAT, MART) and network pruning methods (Hydra, HARP). In Table 3, we use Tiny-ImageNet as our test bed and apply 20G, 40G as model's computation budgets. Then, we apply TRADES, SAT, MART to the teacher network using adversarial training and conduct network pruning to the teacher

Table 2: Baseline Comparison on Tiny-ImageNet

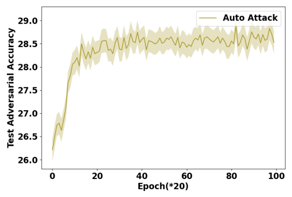
Model	#P(M)	#F(G)	Tiny-ImageNet			
			Clean	PGD^{20}	CW^{40}	AutoAttack
WRN-46-14	128	19.8	41.23±0.05	20.52±0.09	25.78±0.12	16.28±0.11
AdvRush (16@100)	134	22.4	41.77±0.06	20.85±0.12	26.19±0.14	17.05±0.20
RACL (16@108)	133	22.6	41.35±0.07	20.68±0.08	25.93±0.09	16.58±0.17
RobustResNet-A3	75.9	20.1	47.33±0.12	21.50±0.14	28.92±0.15	25.84±0.18
RC-NAS	**72.5**	**18.4**	**48.28±0.17**	**22.79±0.12**	**30.14±0.18**	**28.55±0.14**
WRN-70-16	267	38.7	42.09±0.08	20.68±0.07	26.02±0.05	16.75±0.04
AdvRush (22@100)	266	41.94	42.32±0.09	20.82±0.15	26.74±0.10	17.14±0.18
RACL (22@110)	264	40.88	41.99±0.08	20.74±0.18	26.31±0.12	16.92±0.15
RobustResNet-A4	147	39.2	49.84±0.19	23.38±0.15	30.56±0.20	27.45±0.16
RC-NAS	**145**	**38.9**	**50.15±0.17**	**23.87±0.14**	**31.08±0.24**	**29.64±0.19**

network with score masks (Hydra) or with a holistic aggressive opinion (HARP) to construct baselines. Table 3 clearly shows that the teacher network trained from all non-RL baselines cannot surpass our RL decided sub-network. We further show similar results on cifar datasets in Table 6 of Appendix F.2.

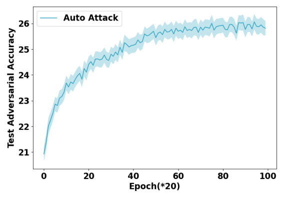
4.2.2 Effectiveness of dual-level training paradigm. Given the effectiveness of RL mechanism, we further investigate whether the novelly designed dual-level training paradigm helps improve the test performance, which includes a meta RL training phase to optimize under different adversarial tasks and a downstream RL fine-tuning phase to let the meta-trained RL agent quickly adapt to the target task setting. We name the RL agent trained directly on the target task setting (without meta training phase) as R-NAS, and the agent trained with the dual-level training paradigm as RC-NAS. Table 3 shows that on a large dataset Tiny-ImageNet, under same computation budgets, RC-NAS consistently improves over R-NAS, as well as enjoys a lower variance because meta training gives RC-NAS a good weight initialization that can be smoothly adapted into any downstream tasks, without suffering the instability resulting

Table 3: RL v.s. net pruning and adversarial training baselines

Category	Training Method	Tiny-ImageNet			
		Clean	*PGD*^{20}	*CW*^{40}	AutoAttack
AT (20G)	TRADES	35.95±0.11	14.12±0.86	16.33±0.15	20.78±0.32
	MART	32.51±0.05	11.87±0.12	15.13±0.08	17.05±0.21
	SAT	31.68±0.06	10.99±0.13	14.43±0.10	16.58±0.25
Network Pruning (20G)	Hydra	35.18±0.08	12.49±0.54	18.59±0.17	17.86±0.43
	HARP	34.77±0.09	11.85±0.72	17.12±0.84	17.08±0.35
RL (20G)	R-NAS	45.12±1.17	17.47±1.24	22.68±1.29	25.94±1.18
	RC-NAS	**48.28±0.17**	**22.79±0.12**	**30.14±0.18**	**28.55±0.14**
AT (40G)	TRADES	36.45±0.10	15.07±0.10	17.09±0.14	22.83±0.34
	MART	32.96±0.08	12.74±0.11	16.05±0.12	19.11±0.28
	SAT	31.98±0.05	11.72±0.12	15.18±0.11	18.53±0.26
Network Pruning (40G)	Hydra	35.92±0.17	13.55±0.65	19.55±0.18	19.94±0.47
	HARP	35.24±0.18	12.93±0.69	18.24±0.99	19.15±0.42
RL (40G)	R-NAS	45.18±1.16	18.49±1.22	23.15±1.30	27.13±1.20
	RC-NAS	**50.15±0.17**	**23.87±0.14**	**31.08±0.24**	**29.64±0.19**



(a) RL downstream fine-tuned



(b) RL downstream trained

Figure 3: Evaluation curves of the RL (w/ and w/o meta RL training) selected sub-networks on target task setting: WRN-46-14 (20G budget) on auto attacked Tiny-ImageNet.

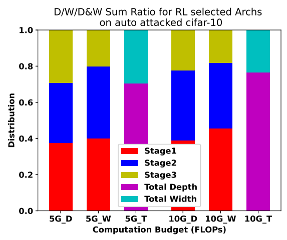
from potentially biased training under a repeated target setting. Such phenomenon has also been verified by the detailed evaluation curves along their respective downstream fine-tuning or training process, where the RL downstream fine-tuned curve with RL meta training will converge fast at earlier training iterations comparing to the RL downstream trained curve w/o meta RL training, shown as Figure 3.

4.2.3 Ablative Study on RL critical design components. We use RL compressed teacher network WRN-46-14 (20G) on Tiny-ImageNet as an example. Given the target task, we first train and downstream fine-tune the RL agent using different design ablation component settings and get the corresponding RL agent with its compressed sub-network. Then, we test the sub-network's classification accuracy on the test set from the same target task, after standard adversarial training. The result is shown in Table 4. We can clearly see that model's performance on all categories will largely drop after the Lipschitz-guided data difficulty embedding *LIPS* being removed in state formulation. Also, without *Cost* which reflects the teacher network's inference costs and capacity, the performance will drop a little around 1%. Annealing represents the model is using soft annealing penalty when constraints are not satisfied, otherwise just assign -1 as a hard penalty for those actions violating the computation budget. By ablating it, the performance also drops around 2% for different attack categories.

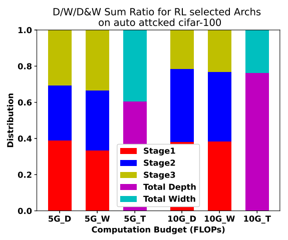
4.2.4 Statistical Analysis. Given different teacher networks (WRN-28-10, WRN-34-12) with computation budgets (5G, 10G), we can analyze the RL agent compress decisions on them for auto attacked

Table 4: Effectiveness of key components

Component			Tiny-ImageNet			
LIPS	*Cost*	Annealing	Clean	*PGD*^{20}	*CW*^{40}	AutoAttack
✗	✓	✓	44.53±0.15	18.78±0.09	27.56±0.18	23.85±0.15
✓	✗	✓	47.45±0.18	21.74±0.11	29.65±0.20	27.92±0.16
✓	✓	✗	46.29±0.18	20.49±0.12	28.74±0.19	26.88±0.15
✓	✓	✓	**48.28±0.17**	**22.79±0.12**	**30.14±0.18**	**28.55±0.14**



(a) Auto attacked CIFAR-10.



(b) Auto attacked CIFAR-100.

Figure 4: RL compressed sub-network statistical analysis: depth/width compression ratio across three stages under 5G and 10G budgets (5G-D, 5G-W, 10G-D, 10G-W), and the total depth to total width compression ratio under 5G (5G-T) and 10G (10G-T) budgets on auto attacked CIFAR 10 and 100.

CIFAR-10 and CIFAR-100 datasets. For WRN-28-10 as teacher network input, it lacks enough generalization ability to auto attack for either CIFAR-10 and CIFAR-100, resulting into a relatively higher width ratio for the last stage comparing to the larger (10G) budget teacher model WRN-34-12, especially for CIFAR-100 which is more complex to classify. For the total width to total depth ratio, 5G model is much higher than 10G model on either CIFAR-10 or CIFAR-100, because the model needs to expand width more to effectively learn the important features for classification given the relatively smaller budget. For the 10G model, it possesses enough parameter learning space and prefers adversarial robustness more than generalization ability, thus tending to reducing the width for higher adversarial robust accuracy. The statistical comparison is shown in Figure 4.

5 CONCLUSION

In this paper, we propose a RC-NAS framework trained by a novel dual-level training paradigm to achieve reinforced compressive neural architecture architecture search. Specifically, given an input teacher network, a dataset and adversarial attack, our RL agent is able to recognize its difficulty level based upon the capacity of the given teacher network and perform the adaptive stage-level and block-level compression to generate a robust sub-network architecture. Experiments show that our proposed RL sub-network achieves an improved test performance on a wide range of target task's test set across different datasets, adversarial attacks and initial teacher networks. We further investigate the topology of the RL generated sub-network to illustrate its effectiveness in selecting a unique and adversarial robust network architecture given the target task requirements. We provide a theoretical justification on why RC-NAS could produce robust architectures through comparison to those standard adversarial training methods.

REFERENCES

[1] Zeyuan Allen-Zhu and Yuanzhi Li. 2022. Feature Purification: How Adversarial Training Performs Robust Deep Learning. arXiv:2005.10190 [cs.LG]

[2] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M. Kitani. 2018. N2N learning: Network to Network Compression via Policy Gradient Reinforcement Learning. In *International Conference on Learning Representations*. https://openreview.net/pdf?id=B1hcZZ-AW

[3] Miriam Bellver, Xavier Giro-i Nieto, Ferran Marques, and Jordi Torres. 2016. Hierarchical Object Detection with Deep Reinforcement Learning. In *Deep Reinforcement Learning Workshop, NIPS*.

[4] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 39–57.

[5] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*. PMLR, 2206–2216.

[6] Chaitanya Devaguptapu, Devansh Agarwal, Gaurav Mittal, Pulkit Gopalani, and Vineeth N Balasubramanian. 2021. On adversarial robustness: A neural architecture search perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 152–161.

[7] Minjing Dong, Yanxi Li, Yunhe Wang, and Chang Xu. 2020. Adversarially robust neural architectures. *arXiv preprint arXiv:2009.00902* (2020).

[8] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[9] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. 2020. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 631–640.

[10] Yiwen Guo, Anbang Yao, and Yurong Chen. 2016. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems* 29 (2016).

[11] Abdul Mueed Hafiz. 2022. Image Classification by Reinforcement Learning With Two-State Q-Learning. *Handbook of Intelligent Computing and Optimization for Sustainable Development* (2022), 171–181.

[12] Hanxun Huang, Yisen Wang, Sarah Erfani, Quanquan Gu, James Bailey, and Xingjun Ma. 2021. Exploring architectural ingredients of adversarially robust deep neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 5545–5559.

[13] Shihua Huang, Zhichao Lu, Kalyanmoy Deb, and Vishnu Naresh Boddeti. 2022. Revisiting Residual Networks for Adversarial Robustness: An Architectural Perspective. *arXiv preprint arXiv:2212.11005* (2022).

[14] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[15] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[16] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[17] Yann LeCun, John Denker, and Sara Solla. 1989. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, D. Touretzky (Ed.), Vol. 2. Morgan-Kaufmann. https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 740–755.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[21] Jisoo Mok, Byunggook Na, Hyeokjun Choe, and Sungroh Yoon. 2021. AdvRush: Searching for adversarially robust neural architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12322–12332.

[22] Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. 2022. Reducing Excessive Margin to Achieve a Better Accuracy vs. Robustness Trade-off. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Azh9QBQ4tR7

[23] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[24] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. 2020. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems* 33 (2020).

[25] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.

[26] Aladin Virmaux and Kevin Scaman. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems* 31 (2018).

[27] Dingrong Wang, Deep Shankar Pandey, Krishna Prasad Neupane, Zhiwei Yu, Ervine Zheng, Zhi Zheng, and Qi Yu. 2023. Deep Temporal Sets with Evidential Reinforced Attentions for Unique Behavioral Pattern Discovery. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 36205–36223. https://proceedings.mlr.press/v202/wang23ab.html

[28] Dingrong Wang, Hitesh Sapkota, Xumin Liu, and Qi Yu. 2021. Deep Reinforced Attention Regression for Partial Sketch Based Image Retrieval. In *2021 IEEE International Conference on Data Mining (ICDM)*. 669–678. https://doi.org/10.1109/ICDM51629.2021.00078

[29] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. 2020. Improving Adversarial Robustness Requires Revisiting Misclassified Examples. In *ICLR*.

[30] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. 2021. Do wider neural networks really help adversarial robustness? *Advances in Neural Information Processing Systems* 34 (2021), 7054–7067.

[31] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. 2020. Adversarial Weight Perturbation Helps Robust Generalization. In *NeurIPS*.

[32] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. 2019. Adversarial robustness vs. model compression, or both?. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 111–120.

[33] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*. PMLR, 7472–7482.

[34] Junzhe Zhang, Daniel Kumor, and Elias Bareinboim. 2020. Causal imitation learning with unobserved confounders. *Advances in neural information processing systems* 33 (2020), 12263–12274.

[35] Xunyu Zhu, Jian Li, Yong Liu, and Weiping Wang. 2023. Improving Differentiable Architecture Search via Self-Distillation. *arXiv preprint arXiv:2302.05629* (2023).

[36] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

Appendix

A ORGANIZATION

We first present all symbols used throughout the paper in Table 5 of Section B. Next, we illustrate the stage encoder separate pre-training process in Figure 5 of Section C as well as describe the detailed meta RL training and downstream fine-tuning process in Algorithm 1 and 2 of Section D. We provide the proof of Lemma 3.1 and Theorem 3.2 in Appendix E. Then, we provide baseline details and additional experiment results in Appendix F, including the RL mechanism F.2 and critical design component F.3 ablative study results, as well as more topology comparison F.4 results on CIFAR-10 and CIFAR-100 datasets under different computation budgets. Finally, we discuss the broader impact of our work in Section G and provide a source code link in section H.

B SUMMARY OF NOTATIONS

Table 5 summarizes the major notations used throughout the paper.

C STATE ENCODER PRE-TRAINING

Figure 5 shows the process of the state encoder training. Specifically, we pre-train it with randomly sampled diverse tasks, where the stage encodings $stage_i$, data difficulty embedding $LIPS$ and inference cost embedding $Cost$ are input into the state encoder to generate the state s_t^i. Then we decode s_t^i into the same inputs again with an additional decoder using the supervision by the input $\texttt{concat}(\texttt{LIPS}, \texttt{ST}_i, \texttt{CT})$ itself. The training target is formulated as Equation (10), where $DE_{\theta_d}(s_t^i)$ means the decoder network.

$$\mathcal{L}_{\text{SE}} = (\texttt{concat}(\texttt{LIPS}, \texttt{ST}_i, \texttt{CT}) - DE_{\theta_d}(SE_\theta(\texttt{LIPS}, \texttt{ST}_i, \texttt{CT})))^2 \tag{10}$$

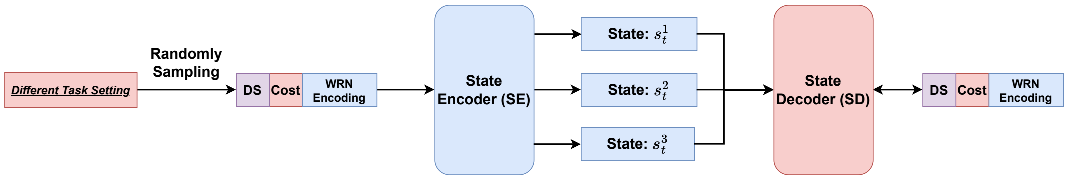


Figure 5: State Encoder Pre-Training

D META RL TRAINING AND DOWNSTREAM RL FINE-TUNING

E THEORETICAL PROOF

In this section, we provide the Proof of Lemma 3.1 and Theorem 3.2. Before formally proving them, first we would define the gradient update in our case which would be the essential block to prove our Lemma and Theorem.

E.1 Gradient Computation

At every iteration t, the gradient of the loss with respect to the neuron Θ_i can be represented as :

$$\Delta_{\Theta_{i,t}} = -yl'(\Theta_{i,t}; \mathbf{x}, y, \rho)\left(\mathbb{1}_{\langle\Theta_{i,t},\mathbf{x}\rangle]+\rho_i\geq b_t} + \mathbb{1}_{-\langle\Theta_{i,t},\mathbf{x}\rangle]+\rho_i\geq b_t}\right)\cdot\mathbf{x} \tag{11}$$

For simplicity, considering the task to be binary classification problem i.e., $y \in [-1, 1]$, and loss to be log-loss represented as:

$$l(\Theta_t; \mathbf{x}, y, \rho) = \log\left(1 + \exp(-yf_t(\Theta; \mathbf{x}, \rho))\right) \tag{12}$$

Further, $l'(\Theta_{i,t}; \mathbf{x}, y, \rho)$ is defined as:

$$l'(\Theta_{i,t}; \mathbf{x}, y, \rho) = \frac{\delta}{\delta s}[\log(1 + \exp(s))]|_{s=-yf_t(\Theta;\mathbf{x},\rho)} \tag{13}$$

E.2 Proof of Lemma 3.1

Suppose at any training iteration t, $\max_{i\in[N]}||v_i||_2 \leq r'$. Also, for the sake of simplicity let $l(\Theta_t; \mathbf{x}, y, \rho) = l \in [-1, 1]$. Also, consider $\delta \in \mathcal{R}^D$ be the perturbation applied on $\mathbf{x}$ with $||\delta||_2 \leq \tau$. Then, l2 norm of average gradient direction bound for $i^{th}; i \in [N]$ neuron in the network with the adversarial input $\mathbf{x} + \delta$ can be represented as:

$$B = ||\mathbb{E}_{\mathbf{x},y,\rho}\left[l\mathbb{1}_{\langle g_i+v_i,\mathbf{x}+\delta\rangle+\rho_i\geq b}(\mathbf{x}+\delta)\right]||_2 \tag{14}$$

Table 5: Summary of Notations

Symbol Group	Notation	Description
Task Related	$f_\psi(\cdot)$	input teacher or generated student network with parameter ψ
	$\mathcal{P}$	adversarial attack method $\mathcal{P}$
	$D, D_{train}, D_{test}, D_{eval}$	dataset, training, test and evalution set, respectively.
	Cost	Inference cost embedding measured by GFLOPs reflecting the teacher network's capacity.
	LIPS	Lipschitz-guided embedding representing adversarial attacked dataset difficulty level to the teacher network.
RC-NAS	SE, DE	State encoder and decoder (for pre-training only).
	θ, θ_d	Parameter space of state encoder and decoder.
	$stage_i, N_{stage}$	ith stage encoding and the number of stages in teacher network.
	H_i^F, H_i^B	Forward and backward LSTM output encodings for $stage_i$ encoding.
	$f_\phi(\cdot)$	Multi-head policy network with its parameter space ϕ.
	$\mathcal{N}(\cdot\|\mu, \sigma^2), Ber(\cdot\|p)$	Gaussian distribution, Bernoulli distribution.
	$\phi_\mathcal{N}, \phi_\mathcal{B}, \phi_\mathcal{E}$	Gaussian head, Bernoulli head and their previous shared feature extraction module's respective parameter space.
	μ_i, Σ_i^2	Output of the Gaussian head of policy network.
	p_i	Output of the Bernoulli head of policy network.
	$\hat{\epsilon}, \hat{\sigma}$	Attack radius (perturbation bound), sigmoid activation function.
	CB, ϵ	Desired computation budget, annealing penalty controlling hyper-parameter when CB is not satisfied.
	ζ_{RL}	Average inference cost of the RL compressed subnetwork on the evaluation set.
	$\tilde{A}_{teacher}, \tilde{A}_{RL}$	Teacher network and RL compressed sub-network's robust accuracy on the task designated evaluation set.
	C	Compression ratio of the RL compressed sub-network compared to its searched teacher network.
	AT	Standard Adversarial Training
Environment	s_t, s_t^i	RL state for the whole network, RL state for stage i at time step t.
	a_t, a_t^i	RL action for compressing whole network, RL action for compressing stage i at time step t.
	$r_t(s_t, a_t)$	RL reward to evaluation the compression effect for adversarial robustness.
	$\mathcal{T}(\cdot\|s_t, a_t)$	RL state transition function, implemented by randomly sampling from three buffers.
Theoretical Results		
	$g_{i,t}$	Pure feature contribution for i^{th} neuron in training iteration t.
	$v_{i,t}$	Dense mixture component for i^{th} neuron in training iteration t.
	ρ_i	ReLU smoothing parameter
	b_t	Bias at the t^{th} training step
	$\mathbf{M}$	Sparse matrix from which input is drawn
	$\Theta_{i,t}$	Hidden weight for i^{th} neuron at time t
	$\mathbf{z}$	Sparse hidden vector with sparsity defined by k
	$\hat{\epsilon}$	Gaussian noise present in the input $\mathbf{x}$
	δ	l2-perturbation applied to the input sample
	τ	Radius defining the l2-perturbation
	S_{RL-C}^A	Compressed student sub-network obtained using RC-NAS and is trained with adversarial loss
	S_U^A	Neural network trained with adversarial loss without RC-NAS
	ΔL_t^{RL-C}	Gradient movement computed in S_{RL-C}^A
	Δ_t^U	Gradient movement computed in S_U^A
	$v_{i,RL-C}^{T+T'}$	Final dense mixture component for the i^{th} neuron of S_{RL-C}^A network
	$v_{i,U}^{T+T'}$	Final dense mixture component for i^{th} neuron of S_U^A network
	T	Total clean training iterations
	T'	Additional Adversarial training performed on the top of clean training

Algorithm 1 Meta RL Training for Compressive Neural Architecture Searching.

```
  Meta RL training:
Require: dataset Buffer $\mathbb{B}_{data}$; Adversarial Attack Buffer $\mathbb{B}_{adv}$; Architecture Buffer $\mathbb{B}_{arch}$; Initialize State Encoder $SE(\cdot|\theta)$, policy Network $f_\phi(\cdot)$, RL total iteration number M, RL total time step T
  for m = 1 to M do
    Initialize architecture buffer $\mathbb{B}_{arch}$ with a pre-defined teacher network pool
    Sample input dataset $D$ from buffer $\mathbb{B}_{data}$
    Sample adversarial attack method $A$ from buffer $\mathbb{B}_{adv}$
    Split $D$ into training $D_{train}$ and evaluation $D_{eval}$, respectively.
    for t = 1 to T do
      Sample a teacher network architecture $Net_{Teach}$ from architecture buffer $\mathbb{B}_{arch}$.
      Train the teacher network on clean training set $D_{train}$ with standard adversarial training $AT$.
      Adversarial attack the teacher network on evaluation set $D_{eval}$ with attack choice $A$ to construct $A$-attacked evaluation set $D_{eval}$.
      Input $D_{eval}, A, Net_{Teach}$ into Stage Encoder $SE(\cdot|\theta)$, encode WRN stages $Stage_i, i \in [1, N_{stage}]$, data difficulty $LIPS$ (1) and inference cost $Cost$ and get the output state $s_t$ using Equation (2).
      Get the action $a_t \sim f_\phi(s_t)$ which designates newly RL compressed network configuration for each stage with Equation (3).
      Generate the RL compressed network $Net_{Stu}$ and store it into the architecture buffer $\mathbb{B}_{arch}$
      Train RL compressed network $Net_{Stu}$ with standard adversarial training $AT$ on clean training set $D_{train}$ and evaluation it on adversarial evaluation set $D_{eval}$ with the same attack choice $A$, the evaluated adversarial robust accuracy $\tilde{A}_{RL}$ is used to form the RL reward $r_t(s_t, a_t)$, with Equation (4).
      Pre-train the state encoder on combined training and evaluation set $D_{train}, Deval$ with Eq. (10), then freeze its weight and optimize the policy network parameters $\phi$ using Eq. (5).
    end for
  end for
  Output: Meta-Trained RL framework, i.e. RL trained state encoder $SE(\cdot|\theta')$ and policy network $f_{\phi'}(\cdot)$.
```

Algorithm 2 Downstream RL fine-tuning and testing for target task setting.

```
Require: Meta-trained State Encoder $SE(\cdot|\theta)$, policy Network $f_\phi(\cdot)$, RL total iteration number $\tilde{M}$, RL total time step $\tilde{T} = 1$, target task setting including dataset ($D_{train}$,$D_{test}$,$D_{eval}$), adversarial attack $A$ and initial teacher network $Net_{Teach}$ for compression.
  Adversarial attack the evaluation set, test set $D_{eval}, D_{test}$ with designated task parameters: Attack choice $A$ targeting standard adversarial trained teacher network $Net_{Teach}$ on training set$D_{train}$.
  Pre-train the state encoder on combined training and evaluation set $D_{train}, Deval$ with target adversarial attack $A$ and initial network $Net_{Teach}$ using Eq. (10), then freeze its weight for later inference use.
  for m = 1 to $\tilde{M}$ do
    for t = 1 to $\tilde{T}$ do
      Input $D_{eval}, A, Net_{Teach}$ into Stage Encoder $SE(\cdot|\theta)$, encode teacher network $Net_{teach}$ stages $Stage_i, i \in [1, N_{stage}]$, data difficulty $LIPS$ (1) and inference cost $Cost$ to get the output state $s_t$ using Equation (2).
      Get the action $a_t \sim P_S(s_t|\phi)$ which designates newly RL compressed network configuration for each stage with Equation (3).
      Generate the RL compressed network $Net_{Stu}$ and store it into the architecture buffer $\mathbb{B}_{arch}$
      Train RL compressed network $Net_{Stu}$ with standard adversarial training on clean training set $D_{train}$ and evaluation it on adversarial evaluation set $D_{eval}$ from the target task setting, the evaluated adversarial robust accuracy $\tilde{A}_{RL}$ is used to form the RL reward $r_t(s_t, a_t)$, with Equation (4).
      Optimize (fine-tuning) the policy network parameters $\phi$ using Eq. (5).
    end for
  end for
  Output: Downstream finetuned RL agent for the target domain task complexity, with trained state encoder $SE(\cdot|\theta')$ and policy network $f_{\phi'}(\cdot)$.
  (Optional) For RL testing:
  Leverage downstream fine-tuned RL agent to sample an RL compressed sub-network from teacher network $Net_{Teach}$, then train it with standard adversarial training on clean training set from the target task, then test it on adversarial attacked test set $D_{test}$ from the same target task setting, for adversarial performance evaluation.
```

It should be noted that for the sake of simplicity, we have removed the subscript t wherever applicable. For the generic gradient direction shown above, we will first derive the upper bound and then instantiate to the RL-guided compressed sub-student network S^A_{RL-C} and dense network without compression i.e., S_U^A. It is worth mentioning that we apply adversarial training on both networks. Using the Analysis of F.4 from [1], we can write the following

$$||\mathbb{E}[l\mathbb{1}_{\langle g_i+v_i,\mathbf{x}+\delta\rangle+\rho_i\geq b}\delta]||_2 \leq \kappa\tau \tag{15}$$

In the above equation $\kappa = O(\frac{k}{D} + \frac{(r')^2}{db^2})$ Now let us try to find the bound for the norm of $\boldsymbol{\beta} = \mathbb{E}[l\mathbb{1}_{\langle g_i+v_i,\mathbf{x}+\delta\rangle+\rho_i\geq b}\mathbf{x}]$

According to Eq. F.7 [1], inner product of $\boldsymbol{\beta}$ with $\mathbf{M}_j; j \in [D]$ can be decomposed into the following three terms:

$$|\langle\boldsymbol{\beta}, \mathbf{M}_j\rangle|| \leq \mathbb{E}\left[\mathbb{1}_{\langle g_i,\mathbf{x}\rangle\geq\frac{b}{10}} + \mathbb{1}_{\langle v_i,\mathbf{x}-\mathbf{M}_j\mathbf{z}_j\rangle\geq\frac{b}{20}}.|\langle\mathbf{x},\mathbf{M}_j|\right] + \frac{1}{poly(D)} \tag{16}$$

The summation of the first term in above equation for all neurons can be represented as (using Eq. F.8 from [1])

$$\sum_{j\in[D]}\left(\mathbb{E}[\mathbb{1}_{\langle g_i,\mathbf{x}\rangle\geq\frac{b}{10}}].|\langle\mathbf{x},\mathbf{M}_j|\right)^2 \leq sO\left(\frac{k}{D^2}\right) \tag{17}$$

It should be noted that the scalar s takes different values depending on the nature of the training. In the case of the RC-NAS student sub-network adversarial training, as the model is forced to zero out the weights, $\langle\mathbf{x},\mathbf{M}_j\rangle$ will always be less compared to the dense network trained using adversarial training. In other words we $u = 1$ in the case of S_U^A, and $0 \leq u \leq 1$ in the case of the S_{RL-C}^A.

Similarly for the second term, applying the fact that $\langle\mathbf{M}_j, \hat{\epsilon}\rangle$ for RC-NAS based student sub-network adversarial training is less than that of the standard adversarial training on the dense network, we have following (according to F.9 from [1]):

$$\sum_{j\in[D]}\left(\mathbb{E}\left[\mathbb{1}_{\langle v_i,\mathbf{x}-\mathbf{M}_j\mathbf{z}_j\geq\frac{b}{20}}.|\langle\mathbf{x},\mathbf{M_j}\rangle|\right]\right)^2 \leq w\left(\frac{(r')^4}{D^2b^4}.\left(\frac{k}{D}+\sigma_x^2\log^2 D\right)\right) \tag{18}$$

Here $w = 1$ for the S_U^A whereas, $0 \leq w \leq 1$ for S_{RL-C}^A.

Similarity the bound for the third term with leveraging the fact that $\langle v_i, \mathbf{M}_j\rangle$ is smaller for the RC-NAS student sub-network compared to the standard adversarial dense network training, we have following (using F.10 in [1])

$$\sum_{j\in[D]}\left([\mathbb{1}_{\langle v_i,\mathbf{M}_j\rangle\mathbf{z}_j\geq\frac{b}{20}}].|\langle\mathbf{x},\mathbf{M}_j\rangle|\right) \leq u\left(\frac{(r')^2}{D^2b^2}\right) \tag{19}$$

In case of S_{RL-C}^A, $0 \leq u \leq 1$ whereas, $u = 1$ for S_U^A. Combining Eqs 17, 18, 19 and 15, we have the following

$$||\mathbb{E}_{\mathbf{x},a,\rho}\left[l\mathbb{1}_{\langle g_i+v_i,\mathbf{x},\delta\rangle+\rho_i\geq b}(\mathbf{x}+\delta)\right]||_2 \leq cO\left(\left(\frac{k}{D}+\frac{(r')^2}{Db^2}\right)\tau+\frac{\sqrt{k}}{d}+\frac{(r')^2}{Db^2}\left(\frac{\sqrt{k}}{\sqrt{D}}+\sigma_x logD\right)+\frac{r'}{Db}\right) \tag{20}$$

In the above equation $c = 1$ for S_U^A whereas $0 \leq c \leq 1$ for the S_{RL-C}^A. Considering left hand side of the above Equation as ΔL_t with ΔL_t^{RL-C} being gradient bound for S_{RL-C}^A and ΔL_t^U being the gradient bound for S_U^A, we can write

$$\Delta L_t^{RL-C} \leq \Delta L_t^U \tag{21}$$

This completes the Lemma 3.1.

E.3 Proof of Theorem 3.2

Considering the Equation 3.1 is true in Lemma 3.1 then the $v_i^{T+T'}$ after T' steps of training can be expressed as

$$||v_i^{T+T'}||_2 \leq ||v_i^T||_2 + T'\eta cO\left(\left(\frac{k}{D}+\frac{(r')^2}{Db^2}\right)\tau+\frac{\sqrt{k}}{D}+\frac{(r')^2}{Db^2}\left(\frac{\sqrt{k}}{\sqrt{D}}+\sigma_x logD\right)+\frac{r'}{Db}\right) \tag{22}$$

Where η is the learning rate used in the SGD to update network. It should be noted that $c = 1$ for the standard dense network adversarial training i.e., on a network S_U^A whereas $0 \leq c \leq 1$ in the RC-NAS guided sparse adversarial training on student sub-network i.e., on S_{RL-C}^A. This means to show $\max_{i\in[N]}||v_i^T + T'|| \leq r'$, we can choose the following

$$cO\left(\left(\frac{k}{D}+\frac{(r')^2}{Db^2}\right)\tau+\frac{\sqrt{k}}{D}+\frac{(r')^2}{Db^2}\left(\frac{\sqrt{k}}{\sqrt{D}}+\sigma_x logD\right)+\frac{r'}{Db}\right) \leq r' \tag{23}$$

Using this we can conclude

$$r'_U \geq r'_{RL-C} \tag{24}$$

Where r'_U is the upper bound of dense mixture component for S_U^A and r'_{RL-C} be the dense mixture upper bound for S_{RL-C}^A. By using this inequality, we can write

$$\max_{i\in N}||v_{i,RL-C}^{T+T'}||_2 \leq \max_{i\in N}||v_{i,U}^{T+T'}||_2 \tag{25}$$

Thie completes the proof for Theorem 3.2.

Table 6: RL v.s. network pruning v.s. adversarial training baselines

Category	Training Method	CIFAR-10				CIFAR-100			
		Clean	PGD^{20}	CW^{40}	AutoAttack	Clean	PGD^{20}	CW^{40}	AutoAttack
AT (5G)	TRADES	84.62±0.06	55.90±0.21	53.15±0.33	51.66±0.29	60.98±0.07	32.79±0.09	27.28±0.12	24.94±0.14
	MART	81.29±0.15	52.85±0.40	51.36±0.33	48.74±0.27	57.29±0.23	29.12±0.25	27.48±0.20	18.94±0.22
	SAT	80.87±0.12	52.44±0.36	50.97±0.09	48.25±0.24	56.88±0.21	28.76±0.22	26.52±0.18	18.23±0.19
Network Pruning (5G)	Hydra	84.14±0.07	53.79±0.18	58.47±0.20	47.15±0.05	60.79±0.08	31.23±0.12	29.82±0.10	21.68±0.05
	HARP	83.12±0.05	52.65±0.08	57.12±0.06	45.98±0.04	59.84±0.09	30.17±0.09	28.95±0.08	20.73±0.07
RL (5G)	R-NAS	83.12±0.23	56.74±0.15	55.69±0.22	54.12±0.23	59.19±0.21	33.42±0.18	27.68±0.16	26.14±0.27
	RC-NAS	**86.32±0.08**	**60.48±0.12**	**58.34±0.25**	**57.66±0.24**	**62.33±0.19**	**34.9±0.15**	**29.95±0.27**	**29.35±0.25**
AT (10G)	TRADES	84.98±0.07	56.45±0.20	53.79±0.35	52.12±0.21	61.96±0.15	33.47±0.18	27.98±0.11	25.89±0.16
	MART	81.98±0.16	53.38±0.25	51.96±0.29	49.42±0.18	57.89±0.29	29.98±0.26	28.15±0.19	19.75±0.23
	SAT	81.88±0.16	53.12±0.38	51.93±0.19	48.98±0.22	57.79±0.22	29.45±0.24	27.31±0.20	18.99±0.23
Network Pruning (10G)	Hydra	84.98±0.15	54.23±0.17	59.19±0.23	47.74±0.06	61.52±0.13	31.85±0.11	30.59±0.11	22.74±0.16
	HARP	83.58±0.06	52.95±0.10	57.74±0.07	46.85±0.12	60.04±0.05	30.54±0.11	29.31±0.12	21.76±0.15
RL (10G)	R-NAS	83.12±0.23	61.74±0.15	56.69±0.22	44.12±0.23	**62.68±0.14**	**36.05±0.19**	**29.13±0.16**	**29.85±0.17**
	RC-NAS	**86.84±0.18**	**61.08±0.35**	**60.45±0.24**	**58.68±0.15**	**63.15±0.22**	**36.96±0.25**	**30.55±0.36**	**30.79±0.33**

Table 7: RC-NAS with different design component ablations under 10G budget.

Component			CIFAR-10				CIFAR-100			
LIPS	*Cost*	Annealing	Clean	PGD^{20}	CW^{40}	AutoAttack	Clean	PGD^{20}	CW^{40}	AutoAttack
✗	✓	✓	80.13±0.14	51.28±0.08	55.98±0.09	44.14±0.12	59.25±0.12	33.85±0.14	27.42±0.18	24.17±0.19
✓	✗	✓	86.42±0.17	54.59±0.19	60.05±0.14	48.29±0.18	63.21±0.18	35.74±0.21	30.13±0.24	26.14±0.32
✓	✓	✗	84.65±0.22	53.12±0.28	58.39±0.26	46.71±0.14	61.32±0.24	34.88±0.22	29.04±0.28	24.95±0.31
✓	✓	✓	**86.84±0.18**	**61.08±0.35**	**60.45±0.24**	**58.68±0.15**	**63.15±0.22**	**36.96±0.25**	**30.55±0.36**	**30.79±0.33**

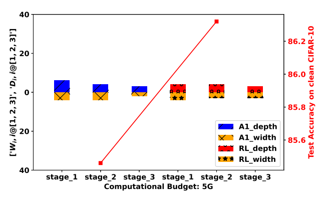
F ADDITIONAL EXPERIMENT

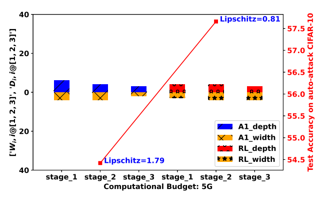
F.1 Baseline Description

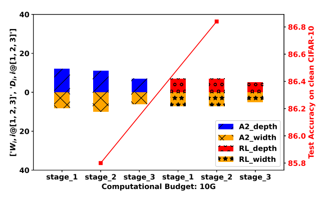
Guo et al. [9] take an architectural perspective and investigate the patterns of network architectures that are resilient to adversarial attacks. To obtain the large number of networks needed for this study, they adopt one-shot neural architecture search, training a large network for once and then finetuning the sub-networks sampled therefrom, the best of the model series searched by this method is called RobNet-large-v2. Dong et al. [7] explore the relationship among adversarial robustness, Lipschitz constant, and architecture parameters and show that an appropriate constraint on architecture parameters could reduce the Lipschitz constant to further improve the robustness, namely RACL. Mok et al. [21] propose AdvRush, a novel adversarial robustness-aware neural architecture search algorithm, based upon a finding that independent of the training method, the intrinsic robustness of a neural network can be represented with the smoothness of its input loss landscape. Through a regularizer that favors a candidate architecture with a smoother input loss landscape, AdvRush successfully discovers an adversarially robust neural architecture. Specifically, we align the network complexity of AdvRush and RACL models by adjusting the number of repetitions of the normal cell N and the input channels of the first normal cell C, denoted as (N@C). Huang et al. [12] propose WRN-34-R based on three key observations derived via a comprehensive investigation on the impact of network width and depth on the robustness of adversarially trained DNNs. Additionally, in the latest work [13], Huang et al. propose a portfolio of adversarially robust residual networks, dubbed RobustResNets A1-A4, spanning a broad spectrum of model FLOP budgets (i.e., 5G - 40G FLOPs), based on a series of architecture search rules found by a large-scale architecture investigation on CIFAR-10 dataset.

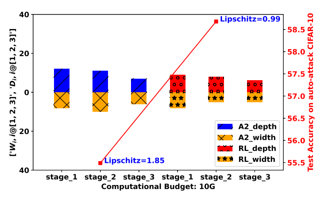
F.2 AT/NP/Meta Ablative Baseline Comparison on cifar datasets

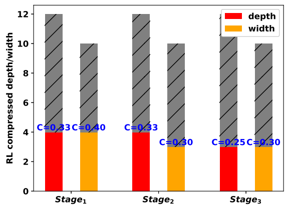
We further collect the adversarial training, network pruning and RL based methods test performance comparison results on a wide range of target task settings, including two cifar datasets, four clean/adversarial attack categories and two initial teacher network with 5G and 10G computation budgets. All non-RL based methods follow the same training and test procedures in the same task setting as compared RL method for fair comparison. We find that with RL dual-level training mechanism, the proposed RC-NAS is consistently better than other non-RL based baselines or the baseline without meta RL training, which aligns with the conclusion in Table 3.

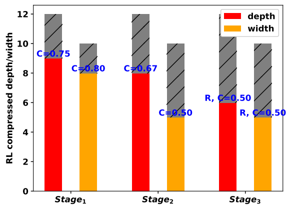

(a) 5G on clean

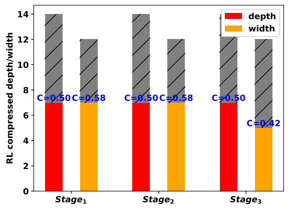

(b) 5G on auto-attack

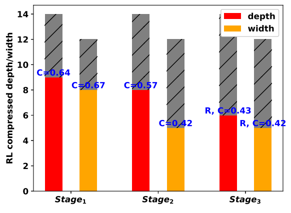

(c) 10G on clean

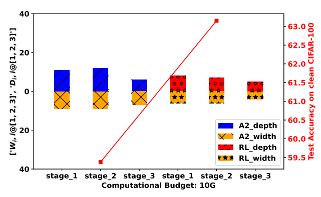

(d) 10G on auto-attack

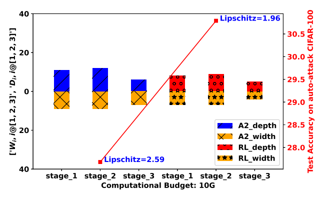

(e) WRN-28-10 on clean

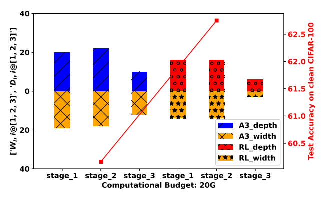

(f) WRN-28-10 on auto-attack

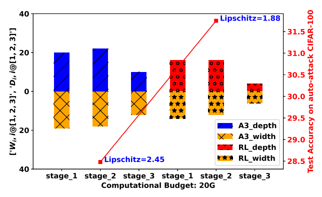

(g) WRN-34-12 on clean

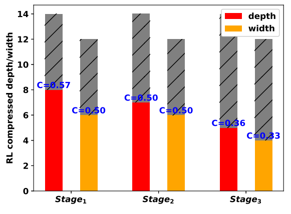

(h) WRN-34-12 on auto-attack

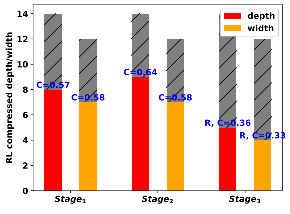

(i) 10G on clean

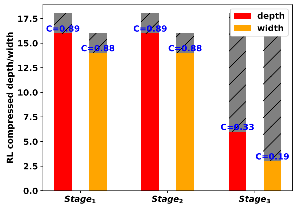

(j) 10G on auto-attack


(k) 20G on clean


(l) 20G on auto-attack


(m) WRN-34-12 on clean


(n) WRN-34-12 on auto-attack


(o) WRN-46-14 on clean

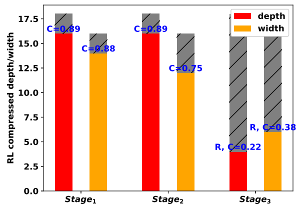

(p) WRN-46-14 on auto-attack

Figure 6: Architecture topology analysis of `RobustResNet` in different attack scenarios from the mildest (*i.e.,* clean) to the most severe (*i.e.,* auto-attack) on CIFAR-10 (a)-(h) and CIFAR-100 (i)-(p) data sets. WRN-28-10, WRN-34-12 and WRN-46-14 are leveraged as teacher networks with 5, 10 and 20 GFLOPs computation budgets, respectively. The corresponding student networks are referred to as `RobustResNet-A1`, `RobustResNet-A2` and `RobustResNet-A3`. The entire teacher network architecture is partitioned into multiple (*e.g.,* 3) stages as in [13] and we visualize both depths and widths of the corresponding `RobustResNet` for each stage. In (a)-(d), the left three bar plots (in blue and orange) show the depths and widths in the three stages of `RobustResNet` A1 and A2 that follow the same configuration rules; the right three bar plots (in red and orange) show the adaptive configuration obtained by the proposed reinforced learning (RL) based architecture search. In (i)-(l), the left three bar plots represent depths and widths in the three stages of `RobustResNet` A2 and A3 and the right three bar plots show the RL compressed network's topology configuration. In (e)-(h) and (m)-(p), the grey bars denote the capacity of the corresponding teacher networks and C denotes the remaining percentage of each stage after compression.

F.3 Additional RL Critical Component Ablation Study

We collect the additional RL designed component ablation study results on CIFAR-10 and CIFAR-100 under 10G computation budget from teacher network WRN-34-12 in Table 7.

F.4 Additional Topology Comparison Results

We collect additional topology comparison results on diverse target task settings: (1) clean CIFAR-10 under 5G/10G budget, (2) auto attacked CIFAR-10 under 5G/10G budget, (3) clean CIFAR-100 under 10G/20G budget and (4) auto attacked CIFAR-100 under 10G/20G budget. The comparison results are shown in Figure 6.

G BROADER IMPACT

Since nowadays data input from real environment is multi-modal, diverse and contain much potential unwanted noises, and most edge computing devices require an efficient memory usage of neural networks, shrinking the network parameter space without harming the generalization and adversarial robustness ability is tempting. Our method provides a novel reinforced compressive architecture search framework which could realize the difficulty level of the input scenarios and take corresponding compression operations to make the sampled neural network architecture more robust to the versatile input challenges after standard adversarial training.

H SOURCE CODE

For the source code, please click here.