ALL-IN-ONE ROBUST MODEL FOR DYNAMIC ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world machine learning systems are increasingly deployed in diverse and dynamic environments, where a model must handle a range of operating conditions and adapt to new ones as they arise. The standard solution, training and maintaining a separate specialist model for each condition, becomes increasingly costly as the number of conditions grows and new conditions continue to appear. We propose the Condition-Aware Network (CAN), which consists of a condition-invariant shared backbone and a lightweight conditional adaptor, allowing a single model to switch flexibly between operating conditions. CAN is trained with a two-stage recipe: joint training over a spectrum of conditions to build a shared representation, followed by fast adaptation that fine-tunes only the lightweight adaptor to a new condition without retraining the backbone. We instantiate CAN primarily in the context of adversarial image classification, and further validate the same conditioning mechanism on image restoration and adversarial text classification, showing that it generalizes across domains with structurally different notions of operating condition. Extensive experiments show that CAN substantially reduces the resources needed to cover multiple conditions while maintaining robustness comparable to a collection of independently trained specialist models. In the CIFAR-10 experiment with a ResNet-18 backbone, CAN reduces training GPU hours by 87.5% and stored parameters by 88.0%, while nearly matching the specialists' robust accuracy. Our code is available at `https://anonymous.4open.science/r/CAN-CF44/`.

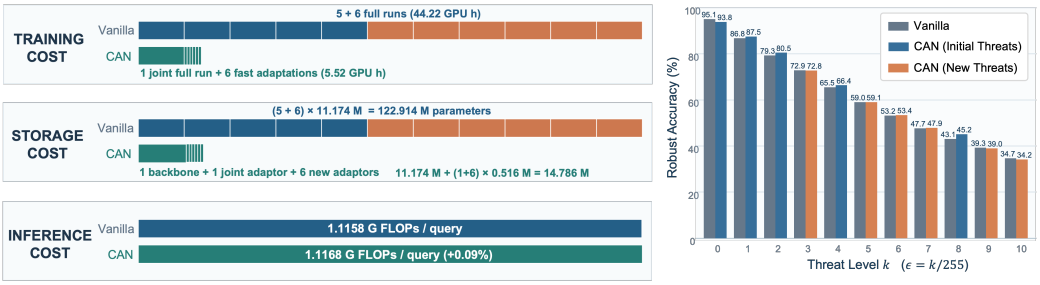
1 INTRODUCTION

Real-world machine learning systems are increasingly deployed in diverse and dynamic environments (Quiñonero-Candela et al., 2008; Koh et al., 2021). This diversity of operating conditions recurs across a wide range of application domains. In system security, a model may receive inputs from a trusted, curated source or from an exposed channel it does not control, such as a retrieval-augmented pipeline pulling content from the open web (Zou et al., 2025), a moderation system scoring user-submitted content (Hosseini et al., 2017), or an agent invoking third-party tools whose outputs it does not control (Greshake et al., 2023). In autonomous driving, a perception system must operate under clear daylight as readily as under heavy rain, fog, or low-light conditions that degrade its sensory inputs (Bijelic et al., 2020; Sakaridis et al., 2018). In image and signal restoration, a model may face anything from mild compression artifacts to severe sensor noise or motion blur, depending on the acquisition conditions of a given input (Zamir et al., 2022; Chiang et al., 2019).

A natural response to this diversity is to maintain a collection of specialists: a separate model specifically optimized for each operating condition of interest. Once the number of conditions is large, this becomes an engineering burden: every new deployment environment demands its own dedicated model, trained and maintained indefinitely alongside all the others. Moreover, whenever a new operating condition arises, for instance content moderation policies tighten, a retrieval pipeline is added, or a sensor is deployed somewhere new, a new specialist must be trained, and the roster of specialists keeps expanding. As modern models grow to millions or billions of parameters, storing and serving a separate, full-sized checkpoint for every operating condition becomes a first-order cost in its own right, and the specialist-per-condition paradigm scales badly against it.

This raises a central practical question: *rather than training and maintaining a collection of models, one per operating condition, can a single model serve every condition along this spectrum, switching freely between them as the environment changes? And when an entirely new condition arises, can that same model be adapted to it quickly, rather than trained again from scratch?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Resource cost comparison.

(b) Performance comparison.

Figure 1: Resource cost and performance comparison for supporting 11 perturbation budgets on CIFAR-10 with a ResNet-18 backbone. The five initial budgets are $\{0, 1, 2, 4, 8\}/255$, and the six new budgets are $\{3, 5, 6, 7, 9, 10\}/255$. **Training:** Vanilla trains 11 specialists for 200 epochs each, requiring 44.22 GPU hours in total, while CAN uses one 200-epoch joint training run plus six 5-epoch adaptor adaptations, requiring 5.52 GPU hours in total, an 87.5% reduction. **Storage:** retaining all 11 conditions requires 122.914M parameters for Vanilla, compared with 14.786M for CAN, i.e., one shared backbone, the jointly trained adaptor, and six adapted adaptors, an 88.0% reduction. **Inference:** for one selected condition, CAN requires 1.1168G FLOPs per query versus 1.1158G for a specialist, a 0.09% increase. **Performance:** despite these large savings in training and storage cost, CAN's robust accuracy closely matches that of Vanilla's dedicated specialists across nearly all threat levels, on both the five initial budgets and the six newly added ones.

To this end, we propose the Condition-Aware Network (CAN), a single framework that spans an entire spectrum of operating conditions rather than committing to one point on it, so that one model can switch freely across severity levels instead of training and maintaining a separate model for each. We instantiate this framework primarily in the context of adversarial robustness and further validate CAN on image restoration and text attacks. These experiments show that the same conditioning mechanism applies across domains with different notions of severity. Our contributions are as follows:

- We develop a general, efficient conditional learning framework that inserts a lightweight adaptor into any backbone. The resulting Condition-Aware Network (CAN) can freely switch between models specialized for different conditions without retraining the backbone. We further analyze how the adaptor affects the model's Lipschitz bound, showing that it modulates input sensitivity only through channel-wise scales.
- We propose a two-stage training procedure. In Stage 1, we jointly train a unified model across a spectrum of condition levels. In Stage 2, we reuse the backbone representation learned in Stage 1 and fast adapt to new conditions by updating only the lightweight adaptor.
- Our extensive experiments show that CAN substantially reduces the resources needed to cover multiple condition levels while maintaining robustness comparable to a collection of independently trained specialist models. In the CIFAR-10 experiment with a ResNet-18 backbone (Figure 1), CAN uses 87.5% fewer training GPU hours and 88.0% fewer stored parameters, while nearly matching the specialists' accuracy. Beyond adversarial image classification, we further validate the generality of CAN through experiments in image restoration and adversarial text classification.

2 Condition-Aware Representation Learning

In this section, we first introduce conditional learning, which yields a Condition-Aware Network (CAN) that adapts a shared backbone to different conditions. We then present a two-stage training recipe that jointly trains CAN across a condition spectrum and efficiently adapts it to new ones.

2.1 CAN: Condition-Aware Network

To adapt a shared backbone to different condition levels, we propose a conditional learning module, illustrated in Figure 3. Given input $\mathbf{x}$ and condition level ϵ, the module intercepts the intermediate backbone representation and applies a lightweight conditional transformation to adapt it to the specified condition level. The architectural details are described below.

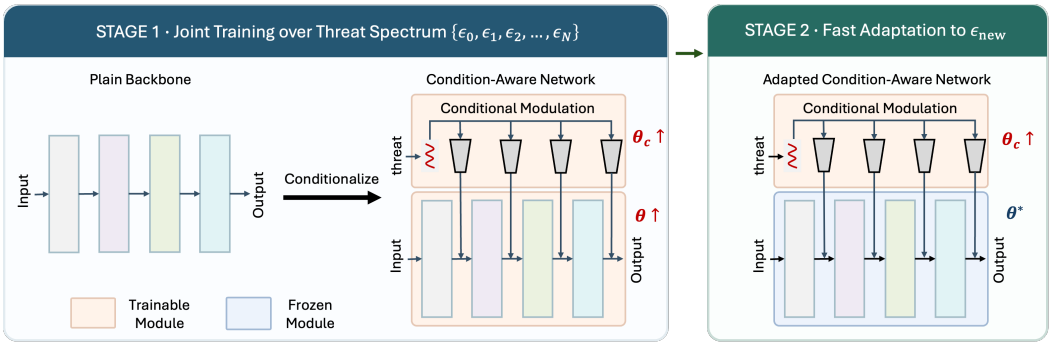


Figure 2: Overview of the Condition-Aware Network (CAN) and its two-stage training recipe. Stage 1 conditionalizes a plain backbone by attaching a conditional adaptor after each intermediate layer, and jointly trains the backbone and adaptor across the condition spectrum $\{\epsilon_0, \epsilon_1, \epsilon_2, \ldots, \epsilon_N\}$. Stage 2 freezes the resulting backbone (blue) and fine-tunes only the lightweight adaptor (red) to quickly adapt the model to a new condition ϵ_{new}, without retraining from scratch.

Condition-level Embedding. To enable expressive conditioning on the scalar ϵ, we adopt a Fourier feature embedding (Rahimi & Recht, 2007):

$$\Phi(\epsilon) = [\sin(2\pi\boldsymbol{\omega}\epsilon), \cos(2\pi\boldsymbol{\omega}\epsilon)] \in \mathbb{R}^d, \ \boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_{d/2}).$$

This embedding maps ϵ into a high-dimensional representation space, allowing the model to approximate complex, non-linear dependencies on the perturbation level. From a kernel perspective, this can be viewed as enabling a rich class of stationary functions over ϵ, facilitating smooth interpolation across various perturbation strengths.

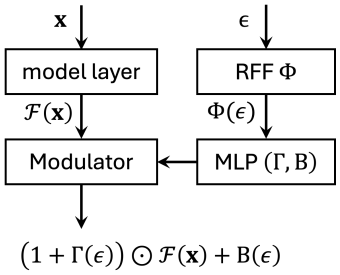


Figure 3: Condition-aware learning.

Channel-wise Conditional Modulation. Given the embedding $\Phi(\epsilon)$, we generate conditioning parameters through a lightweight network, the conditional adaptor, with parameters θ_c: $\big(\Gamma(\epsilon), B(\epsilon)\big) = \mathrm{MLP}_{\theta_c}\big(\Phi(\epsilon)\big)$. These parameters are then injected into intermediate feature activations through a channel-wise affine modulation, as inspired by Perez et al. (2018). Formally, let $\mathbb{H} = \{\mathcal{F} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^d\}$ denote the space $\mathbb{H}$ of candidate backbone functions. The conditional adaptor acts on the function space by transforming intermediate feature representations as follows:

$$\mathcal{T}(\epsilon)[\mathcal{F}] : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}, \quad \mathbf{x} \mapsto (1 + \Gamma(\epsilon)) \odot \mathcal{F}(\mathbf{x}) + \mathrm{B}(\epsilon), \tag{1}$$

for any $\mathcal{F} \in \mathbb{H}$, where $\mathcal{F}(\mathbf{x})$ denotes an intermediate feature tensor and $\odot$ denotes element-wise multiplication. In particular, applying $\mathcal{T}(\epsilon)$ to the backbone $\mathcal{F}$, with parameters θ, gives the conditional model $\mathcal{T}(\epsilon)[\mathcal{F}]$. Because the adaptor is defined as an operator on $\mathbb{H}$ rather than tied to any particular backbone, the same construction applies regardless of the choice of $\mathcal{F}$. This mechanism allows the same backbone network to realize different feature geometries under different condition levels, without introducing separate models for each condition budget. Compared with training multiple independent robust classifiers, such conditional modulation offers a more parameter-efficient way to encode perturbation-specific behavior while preserving shared semantic structure across regimes. The conditional operator $\mathcal{T}(\epsilon)$ can be inserted after any intermediate layer of a given backbone.

2.2 Two-Stage Training Recipe

Having defined the CAN architecture, we now turn to how it is trained and deployed, following a two-stage recipe. **Stage 1** trains the model to be robust across the full range of conditions it is expected to encounter at deployment, without committing to any single operating condition. **Stage 2** addresses a complementary, more practical concern: what happens when the deployed model later faces a condition outside that original range. Rather than retraining the entire model from scratch, the second stage reuses the representation learned in Stage 1 and adapts only the lightweight conditioning mechanism to the new condition.

Stage 1: Joint Training over Condition Spectrum (tune both the backbone θ and the adaptor θ_c). Because CAN targets a single model that operates across a range of conditions rather than one

fixed regime, we train it with respect to a distribution over conditions rather than a single operating point. Let $\mathcal{P}$ be a distribution over condition levels ϵ, and let $\mathbf{x}_\epsilon$ denote the input as observed under condition ϵ (e.g., an image degraded by weather or low-light corruption of severity ϵ, or a text example perturbed at strength ϵ). We train CAN by jointly optimizing the backbone parameters θ and the conditional adaptor parameters θ_c: $\min_{\theta,\,\theta_c} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathbb{E}_{\epsilon\sim\mathcal{P}}\left[\mathcal{L}\Big(\mathcal{T}_{\theta_c}(\epsilon)[\mathcal{F}_\theta]\big(\mathbf{x}_\epsilon\big), y\Big)\right]\right]$, where $\mathcal{L}$ is the task loss (e.g., a classification or reconstruction loss). By sampling ϵ from $\mathcal{P}$ during training, CAN learns an adaptive conditional decision rule that remains effective across the sampled range of operating conditions, instead of overfitting to one particular condition.

Stage 2: Fast Adaptation to New Condition (freeze the backbone θ, tune only the adaptor θ_c). Stage 1 makes CAN effective across the conditions represented in $\mathcal{P}$, but a deployed model may later encounter an operating condition ϵ_{new} outside this range. Retraining the entire model for every such shift would reintroduce the very scalability problem CAN is meant to avoid. Instead, we exploit the structure of the conditional operator $\mathcal{T}_{\theta_c}(\epsilon)$: the backbone already provides a general feature representation shared across all conditions, and only the adaptor that produces $(\Gamma(\epsilon), B(\epsilon))$ needs to specialize to the new condition. Let θ^* denote the backbone parameters obtained at the end of Stage 1. We freeze the backbone at $\mathcal{F}_{\theta^*}$ and fine-tune only the adaptor, with parameters θ_c, by solving $\min_{\theta_c} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathcal{L}\Big(\mathcal{T}_{\theta_c}(\epsilon_{\text{new}})[\mathcal{F}_{\theta^*}]\big(\mathbf{x}_{\epsilon_{\text{new}}}\big), y\Big)\right]$, for a small number of epochs. Because we reuse the well-trained backbone from Stage 1 and fine-tune only the lightweight adaptor, Stage 2 is substantially cheaper than Stage 1, allowing CAN to be extended to a previously unseen condition quickly, without retraining the model from scratch.

2.3 Lipschitz Analysis of the Conditional Adaptor

We analyze how the conditional adaptor $\mathcal{T}(\epsilon)$ affects the input sensitivity of the model. We view this as an analysis of the mechanism rather than a certified defense: a Lipschitz bound alone does not imply robustness, since it also scales with the magnitude of the logits. Let the backbone be a composition of L blocks, $\mathcal{F}_\theta = \mathcal{F}_L \circ \cdots \circ \mathcal{F}_1$ with $\mathcal{F}_\ell \in \mathbb{H}$. The adaptor is applied after each block, giving the conditioned model

$$g_\epsilon := \mathcal{T}_L(\epsilon)[\mathcal{F}_L] \circ \cdots \circ \mathcal{T}_1(\epsilon)[\mathcal{F}_1], \qquad \mathcal{T}_\ell(\epsilon)[\mathcal{F}_\ell](\mathbf{h}) = \big(1 + \Gamma_\ell(\epsilon)\big) \odot \mathcal{F}_\ell(\mathbf{h}) + \mathrm{B}_\ell(\epsilon), \quad (2)$$

where $(\Gamma_\ell, \mathrm{B}_\ell)$ depend on ϵ only and not on the input. The case $L = 1$ recovers $\mathcal{T}(\epsilon)[\mathcal{F}_\theta]$.

Assumption 1. Each block $\mathcal{F}_\ell$ is Λ_ℓ-Lipschitz with respect to the ℓ_2 norm, i.e., $\|\mathcal{F}_\ell(\mathbf{h}) - \mathcal{F}_\ell(\mathbf{h}')\|_2 \le \Lambda_\ell \|\mathbf{h} - \mathbf{h}'\|_2$. This holds for standard convolutional or linear layers with inference-mode normalization and 1-Lipschitz activations (ReLU/GELU), where Λ_ℓ is bounded by the operator norm of the layer's linear map (for convolutions, of the full convolution operator rather than the reshaped kernel) times the Lipschitz constant of the normalization (Szegedy et al., 2013).

Theorem 1 (Lipschitz bound of the conditioned model)**.** *Under Assumption 1, for any condition level ϵ the conditioned model g_ϵ is Lipschitz continuous with respect to the ℓ_2 norm, with*

$$\mathrm{Lip}(g_\epsilon) \;\le\; \bar{\Lambda}(\epsilon) := \prod_{\ell=1}^{L} s_\ell(\epsilon)\, \Lambda_\ell, \qquad s_\ell(\epsilon) := \|1 + \Gamma_\ell(\epsilon)\|_\infty. \quad (3)$$

For $L = 1$ this reads $\|1 + \Gamma(\epsilon)\|_\infty \cdot \Lambda_{\mathcal{F}}$. The proof is given in Appendix A.3.

Theorem 1 shows that the adaptor affects the Lipschitz bound of the model only through the channel-wise scales $s_\ell(\epsilon)$, while the shared block constants Λ_ℓ are unchanged. For reference, Table 8 reports the learned $\Gamma_\ell(\epsilon)$ under different conditions.

3 Experiments

In this section, we conduct a comprehensive evaluation of our method, including one-for-all robustness, overall performance comparison, transferability analysis, and ablation studies.

3.1 Experimental Settings

Datasets and Models. For adversarial robustness, we evaluate CAN on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) using ResNet-18 (He et al., 2016) and WideResNet-28-10 (Zagoruyko & Komodakis, 2016) as backbones, with ResNet-18 as the default unless otherwise specified. For image

restoration, we evaluate four single-weather restoration tasks from WeatherBench (Guan et al., 2025) and LOL-v2 Yang et al. (2021), using 200 test images per task (800 images in total) and a NAFNet-32 backbone (Chen et al., 2022). For text classification, we use backbone model BERT (Devlin et al., 2019) on AG News (Zhang et al., 2015) under TextFooler attacks (Jin et al., 2020) with varying similarity thresholds. For adversarial image and text classification, the conditioning variable is a scalar attack strength, whereas for image restoration, the four degradation types (rain, fog, snow, and low light) are encoded as categorical IDs 1, 2, 3, and 4, respectively.

Baselines and Evaluation. We compare against several groups of baselines: (1) single-model AT baselines, including vanilla PGD-AT (Madry et al., 2017) trained with both fixed and uniform perturbation budgets, as well as representative adversarial training methods TRADES (Zhang et al., 2019), MART (Wang et al., 2019), and NuAT (Sriramanan et al., 2021); and (2) multi-model baselines, including MoRE (Cheng et al., 2021) and ensemble models. We evaluate robustness under a diverse set of adversarial attacks, including FGSM (Goodfellow et al., 2014), PGD (Madry et al., 2017), and AutoAttack (Croce & Hein, 2020). We use PGD-20 as the default attack unless otherwise specified.

Training Details. In Stage 1, all models are trained for 200 epochs with a batch size of 128, momentum of 0.9, and weight decay of 2×10^{-5}, using SGD with an initial learning rate of 0.1, decayed by a factor of 10 at epochs 100 and 150. We model the training condition distribution $\mathcal{P}$ as a *clean-biased uniform distribution*: a mixture that places elevated probability mass on the clean budget ϵ_0 while distributing the remainder uniformly across the remaining condition levels, $\mathcal{P} = p \cdot \delta_{\epsilon_0} + (1-p)\,\mathrm{Unif}\{\epsilon_1, \ldots, \epsilon_N\}$, where $p = \frac{M}{M+N}$, $N+1$ is the total number of condition levels under consideration, and M is a hyperparameter governing the relative sampling weight assigned to clean examples; larger M increasingly biases training toward the clean regime. We provide an ablation on p in Appendix B.3. In Stage 2, we freeze the backbone and fine-tune only the noise-conditioning MLP for 5 epochs with a constant learning rate of 10^{-3}.

3.2 One-for-All Model Across Operating Conditions

We evaluate CAN as a single, unified model across three distinct operating scenarios, (1) adversarial image classification, (2) image restoration, and (3) adversarial text classification, testing whether the same conditional mechanism generalizes across structurally different notions of operating condition.

Table 1: Accuracy (%) across adversarial training and test budgets on CIFAR-10 and CIFAR-100. Gray cells mark matched training and test budgets for AT; darker cells mark CAN.

	Backbone	**ResNet-18**					**WideResNet-28-10**				
Dataset	**Method \ Budget**	$\frac{0}{255}$	$\frac{1}{255}$	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$	$\frac{0}{255}$	$\frac{1}{255}$	$\frac{2}{255}$	$\frac{4}{255}$	$\frac{8}{255}$
CIFAR-10	AT w/ $\epsilon = \frac{0}{255}$	95.09	57.86	8.94	0.14	0.00	95.37	50.41	5.44	0.00	0.00
	AT w/ $\epsilon = \frac{1}{255}$	93.09	86.80	76.80	51.01	20.50	94.67	88.55	78.66	50.20	14.12
	AT w/ $\epsilon = \frac{2}{255}$	91.11	85.81	79.28	61.39	28.68	93.22	88.50	81.91	63.93	28.29
	AT w/ $\epsilon = \frac{4}{255}$	88.51	83.92	78.61	65.51	36.57	90.91	86.72	81.90	68.66	39.79
	AT w/ $\epsilon = \frac{8}{255}$	84.07	80.27	75.82	65.71	43.10	86.40	82.57	78.13	68.15	45.09
	CAN (Ours)	93.85	87.46	80.49	66.36	45.16	94.56	88.49	82.32	68.87	46.63
CIFAR-100	AT w/ $\epsilon = \frac{0}{255}$	75.42	32.27	5.20	0.27	0.01	78.38	26.28	3.31	0.11	0.00
	AT w/ $\epsilon = \frac{1}{255}$	69.83	57.94	44.98	24.15	5.05	74.54	61.34	47.99	23.62	4.41
	AT w/ $\epsilon = \frac{2}{255}$	66.11	57.53	48.79	31.39	10.76	71.43	61.99	51.73	33.02	10.85
	AT w/ $\epsilon = \frac{4}{255}$	61.08	54.08	47.07	34.30	15.93	67.18	60.18	53.32	39.41	18.32
	AT w/ $\epsilon = \frac{8}{255}$	55.62	50.18	44.94	35.20	20.10	60.24	54.80	49.09	39.24	22.95
	CAN (Ours)	70.71	55.91	47.01	38.86	20.34	75.42	62.73	52.12	37.76	22.21

Adversarial Image Classification. We evaluate CAN on CIFAR-10 and CIFAR-100 using ResNet-18 and WideResNet-28-10, as shown in Table 1. We consider the following set of condition levels: $\{\frac{0}{255}, \frac{1}{255}, \frac{2}{255}, \frac{4}{255}, \frac{8}{255}\}$. For each vanilla adversarial training (AT) baseline, a single fixed budget from this set is used for training. For CAN, we instead apply distributional adversarial training over the full set of condition levels. From these results, we draw the following observations: (1) Performance conflict in separately trained models. Models trained at different perturbation budgets are highly specialized: each performs best on its matched test budget (diagonal entries), but degrades significantly under mismatched settings. This reveals poor cross-budget generalization. (2) CAN

approximates the optimal diagonal. CAN achieves performance consistently close to the best diagonal entries using a single conditional model, effectively recovering multiple specialized behaviors.

Image Restoration. To evaluate CAN beyond adversarial robustness, we use NAFNet as the backbone for single-image restoration across four degradation types: rain, fog, snow, and low light. Each specialist is trained on one degradation, whereas CAN conditions a shared model on the degradation type. Table 2 shows that specialists perform well on their matched degradations but often transfer poorly to others. For example, the rain specialist achieves 33.48 dB on rain but only 11.45 dB on low light. Using a single model, CAN comes within 0.58 dB of the rain specialist and 1.43 dB of the snow specialist on their matched conditions, while surpassing the fog and low-light specialists by 1.58 dB and 0.58 dB, respectively. These results suggest that conditional adaptation can extend beyond perturbation budgets to distinct image degradations. We leave visualizations of the restored images to Appendix 7.

Table 2: Image restoration with a NAFNet backbone. Entries report full-image PSNR (dB; higher is better). Each specialist is trained on one degradation; CAN uses a single unified model.

Method \ Degradation	Rain	Fog	Snow	Low-light
Rain specialist	33.48	14.06	23.07	11.45
Fog specialist	23.27	19.06	20.95	13.26
Snow specialist	27.63	13.56	27.66	11.42
Low-light specialist	13.13	10.54	14.80	23.16
CAN	32.90	20.64	26.23	23.74

Adversarial Text Classification. Beyond visual classification, we further validate our method in the language domain on AG News using a BERT-based classifier under TextFooler attacks with different tolerated word embedding difference thresholds ϵ. Our conditional method achieves stronger overall performance across different similarity levels, showing that the proposed framework generalizes beyond image classification. Notably, even when conditioned on a single fixed level, our model still maintains competitive performance across a wide range of perturbation strengths, further demonstrating its transferability and supporting the view that robustness is better modeled as a continuous spectrum rather than a collection of isolated training objectives.

Table 3: Performance on AG News under TextFooler with varying similarity thresholds ϵ. CAN achieves most balanced clean-robust performance across perturbation levels than AT.

Method \ Threshold ϵ	0.0	0.1	0.3	0.5	0.7	0.9
AT, $\epsilon = 0.0$	94.2	82.0	40.0	17.3	8.4	7.9
AT, $\epsilon = 0.1$	94.1	82.5	39.3	16.3	7.8	7.0
AT, $\epsilon = 0.3$	94.5	87.0	45.9	19.4	8.4	8.2
AT, $\epsilon = 0.5$	94.3	87.1	47.9	29.6	15.7	14.7
AT, $\epsilon = 0.7$	93.4	85.3	48.3	28.4	17.5	17.2
AT, $\epsilon = 0.9$	93.1	83.9	50.2	33.9	23.5	23.3
CAN	94.1	86.0	53.1	31.5	20.4	19.8

Table 4: Accuracy (%) under PGD-20 on CIFAR-10 with ResNet-18 across five evaluation budgets. Condition-Blind CAN uses a fixed conditioning value of $\epsilon = 8/255$ for every budget; Condition-Matched CAN is given the evaluation budget. "Sum" adds accuracy across the five budgets, and # Param. reports model parameters.

Method	**Evaluation budget ϵ**					**Sum**	**# Param.**
	$\mathbf{0/255}$	$\mathbf{1/255}$	$\mathbf{2/255}$	$\mathbf{4/255}$	$\mathbf{8/255}$		
Natural Training	95.09	57.86	8.94	0.14	0.00	162.03	11.17M
PGD-AT ($\epsilon = 8/255$)	84.07	80.27	75.82	65.71	43.10	348.97	11.17M
PGD-AT (Uniform ϵ)	87.59	83.78	78.58	67.01	39.21	356.17	11.17M
TRADES ($\lambda = 2.0$)	81.92	78.00	73.51	62.89	41.50	337.82	11.17M
TRADES ($\lambda = 0.2$)	91.49	62.74	27.58	10.27	3.31	195.39	11.17M
MART ($\lambda = 5.0$)	80.92	77.36	73.46	64.57	44.85	341.16	11.17M
MART ($\lambda = 0.5$)	82.29	78.18	73.66	63.56	42.25	339.94	11.17M
NuAT	80.23	77.61	75.10	69.13	46.18	348.25	11.17M
MoRE	84.52	77.10	72.80	63.10	41.22	338.74	55.87M
Ensemble	94.46	87.27	73.95	49.89	20.13	325.70	55.87M
CAN (Condition-Blind)	87.44	83.73	79.58	69.80	45.16	<u>365.71</u>	11.69M
CAN (Condition-Matched)	93.85	87.46	80.49	66.36	45.16	**373.32**	11.69M

3.3 State-of-the-art Performance

We compare CAN with natural training, fixed- and uniformly sampled-budget PGD-AT, TRADES, MART, NuAT, and two multi-model baselines, MoRE and an ensemble. All methods are evaluated on CIFAR-10 with ResNet-18 under PGD-20 at five perturbation budgets. We consider two inference settings. When the evaluation budget is known, CAN (Condition-Matched) uses it as the conditioning

input. When the budget is unknown, CAN (Condition-Blind) uses $\epsilon = 8/255$, the largest budget considered, as a fixed conditioning input for every example. As shown in Table 4, Condition-Matched CAN achieves the highest summed accuracy, 373.32, with 11.69M parameters. Even without access to the evaluation budget, Condition-Blind CAN achieves a summed accuracy of 365.71, outperforming every baseline on this aggregate measure, including uniformly sampled-budget PGD-AT (356.17) and fixed-budget PGD-AT at $8/255$ (348.97). This result is consistent with the cross-budget transferability of the shared representation: conditioning on a fixed budget retains strong performance across the budgets, while budget-matched conditioning provides a further gain when the budget is available.

3.4 Transferability Analysis

To further evaluate the generalization ability of CAN across perturbation levels, we study its *transferability* under mismatched conditioning. We conduct two complementary analyses, including performance comparison and representation visualization.

Performance Transferability. In practice, the adversarial condition level is often unknown or dynamically changing at test time. It is therefore important to evaluate how well models transfer under mismatched perturbation budgets. Vanilla AT trains a separate model $\mathcal{F}_{\theta_i^*}$ for each budget ϵ_i, while CAN uses a single backbone $\mathcal{F}_{\theta^*}$ trained with a uniform distribution over budgets, adapting via the conditional operator $\mathcal{T}(\epsilon_i)[\cdot]$. For a fair comparison, we evaluate each AT specialist against the CAN variant conditioned on the same budget. As shown in Table 5, CAN achieves higher average accuracy in every row, demonstrating consistently superior cross-budget transferability. This stems from learning a shared representation across perturbation levels, which enforces smoothness across ϵ and preserves strong performance even under mismatched test conditions where independently trained AT models degrade substantially.

Table 5: Transfer performance under mismatched conditions on CIFAR-10 with ResNet-18. CAN conditioned on ϵ_i outperforms the AT model separately trained on ϵ_i in every row-wise comparison.

Model \ Budget		ϵ_0	ϵ_1	ϵ_2	ϵ_3	ϵ_4	Avg.
AT	$\mathcal{F}_{\theta_0^*}$	95.09	57.86	8.94	0.14	0.00	32.41
	$\mathcal{F}_{\theta_1^*}$	93.09	86.80	76.80	51.01	20.50	65.64
	$\mathcal{F}_{\theta_2^*}$	91.11	85.81	79.28	61.39	28.68	69.25
	$\mathcal{F}_{\theta_3^*}$	88.51	83.92	78.61	65.51	36.57	70.62
	$\mathcal{F}_{\theta_4^*}$	84.07	80.27	75.82	65.71	43.10	69.79
CAN	$\mathcal{T}(\epsilon_0)[\mathcal{F}_{\theta^*}]$	93.85	86.60	74.50	44.93	11.68	62.31
	$\mathcal{T}(\epsilon_1)[\mathcal{F}_{\theta^*}]$	92.86	87.46	79.48	58.06	22.16	68.00
	$\mathcal{T}(\epsilon_2)[\mathcal{F}_{\theta^*}]$	92.26	87.47	80.49	62.26	26.57	69.81
	$\mathcal{T}(\epsilon_3)[\mathcal{F}_{\theta^*}]$	90.96	86.80	81.03	66.36	33.91	71.81
	$\mathcal{T}(\epsilon_4)[\mathcal{F}_{\theta^*}]$	87.44	83.73	79.58	69.80	45.16	73.14

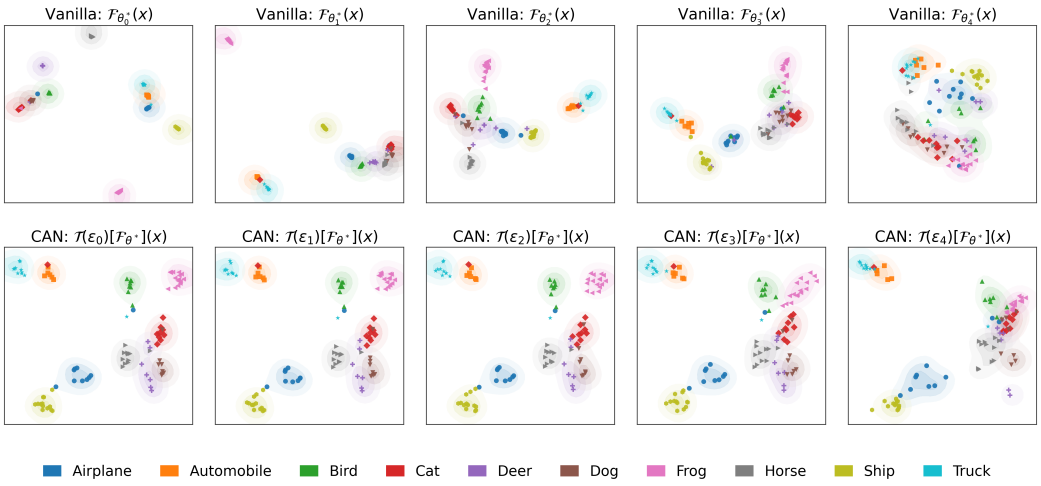


Figure 4: UMAP visualization of learned representations under different adversarial levels. **Top row**: independently trained vanilla models at different noise levels. **Bottom row**: CAN conditioned on different adversarial levels. Under mismatched conditions, vanilla representations degrade and fragment, whereas CAN preserves a coherent global structure, demonstrating stronger transferability.

Visualization. Figure 4 visualizes UMAP embeddings of the final hidden representations under mismatched adversarial conditions, using the same setup as "Performance Transferability." Under matched conditions, both vanilla adversarial training and CAN produce well-separated class clusters. Under mismatched conditions, however, vanilla representations progressively fragment and lose discriminability, whereas CAN preserves a coherent global structure across adversarial levels. This

indicates that CAN learns a family of representations parameterized by adversarial level, explaining its stronger transferability. Embedding difference analysis is provided in AppendixB.2.

3.5 Additional Study

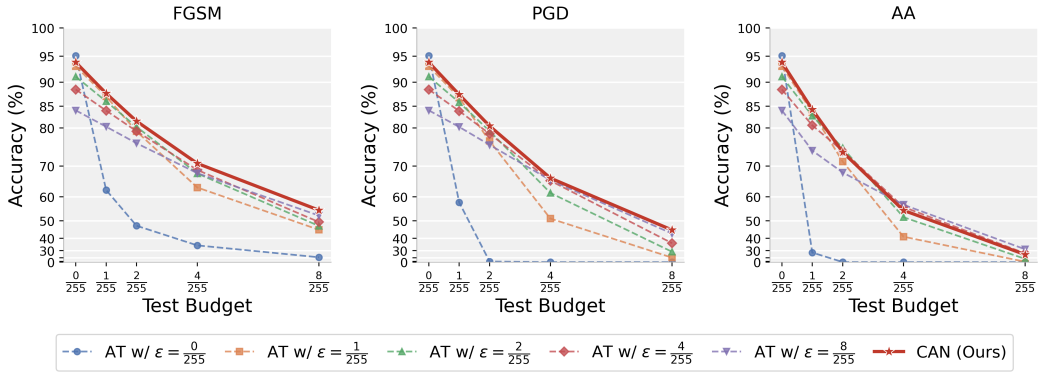


Figure 5: Robustness against FGSM, PGD, and AutoAttack (AA) across varying test budgets.

Different Attacks. We compare CAN against adversarially trained (AT) models, each specialized for a fixed training budget $\epsilon \in \{0, \frac{1}{255}, \frac{2}{255}, \frac{4}{255}, \frac{8}{255}\}$, under three attack types (FGSM, PGD, and AutoAttack) across varying test budgets (Figure 5). The experiments are performed on CIFAR-10 with ResNet18. As shown, AT models tend to overfit to their specific training budget, performing well near their trained ϵ but degrading significantly at other test budgets. In contrast, CAN mostly matches the best-performing AT baseline across all test budgets with one single model.

Fast Adaptation to New Conditions. We next test Stage 2, where CAN adapts to a new condition level by freezing the backbone learned in Stage 1 and fine-tuning only the lightweight adaptor, rather than training a new specialist from scratch. We treat $\epsilon \in \{\frac{3}{255}, \frac{5}{255}, \frac{6}{255}, \frac{7}{255}, \frac{9}{255}, \frac{10}{255}\}$, the budgets outside CAN's Stage 1 training set $\{\frac{0}{255}, \frac{1}{255}, \frac{2}{255}, \frac{4}{255}, \frac{8}{255}\}$, as new condition levels, and compare a dedicated AT specialist at each budget against CAN both before and after the Stage 2 adaptation step. Before adaptation, CAN already tracks the specialist reasonably well within its Stage 1 range, reaching 72.21%, 56.26%, 51.14%, and 41.54% at $\frac{3}{255}$, $\frac{5}{255}$, $\frac{6}{255}$, and $\frac{7}{255}$, but collapses to 0.59% and 1.57% at $\frac{9}{255}$ and $\frac{10}{255}$, both beyond the largest trained budget, a gap of over 33 points from the matched specialists. After adaptation, CAN recovers at every budget, reaching 38.95% and 34.17% at $\frac{9}{255}$ and $\frac{10}{255}$ and tracking the specialist within 0.25 points on average across all six budgets, matching or exceeding it at three of them ($\frac{5}{255}$, $\frac{6}{255}$, and $\frac{7}{255}$), for instance 53.39% vs. 47.89% at $\frac{6}{255}$ and $\frac{7}{255}$ against 53.19% and 47.67% for the specialists. This shows the lightweight adaptation step is essential rather than incidental: it is what lets CAN reach specialist-level robustness once the new condition falls outside the range it can already interpolate.

Table 6: Fast adaptation to new condition levels. CAN can almost match each retrained specialist after a lightweight adaptation step.

Method \ Budget	$\frac{3}{255}$	$\frac{5}{255}$	$\frac{6}{255}$	$\frac{7}{255}$	$\frac{9}{255}$	$\frac{10}{255}$
AT w/ $\epsilon = \frac{3}{255}$	72.87	56.65	48.62	40.40	28.21	23.40
AT w/ $\epsilon = \frac{5}{255}$	72.55	59.04	52.14	45.58	33.55	28.33
AT w/ $\epsilon = \frac{6}{255}$	71.66	59.40	53.19	46.68	35.30	30.13
AT w/ $\epsilon = \frac{7}{255}$	71.20	59.80	53.66	47.67	36.73	32.02
AT w/ $\epsilon = \frac{9}{255}$	70.41	60.07	54.80	49.16	39.33	34.54
AT w/ $\epsilon = \frac{10}{255}$	69.72	59.55	54.63	49.36	38.78	34.70
CAN before Adaptation	72.21	56.26	51.14	41.54	0.59	1.57
CAN after Adaptation	72.76	59.08	53.39	47.89	38.95	34.17

Table 7: Ablation study on condition-level embeddings.

	Discrete	Continuous	Fourier	**0.0**	**0.05**	**0.1**	**0.15**	**0.2**	**0.25**	**0.3**
CAN	✓			82.5	75.1	63.2	48.5	33.0	18.3	5.8
		✓		86.4	76.6	62.7	44.0	28.1	14.2	3.5
		✓	✓	90.4	77.4	65.4	51.7	38.5	25.8	14.1
Baseline	AT w/ $\epsilon = 0.0$			91.2	49.9	7.4	0.3	0.0	0.0	0.0
	AT w/ $\epsilon = 0.3$			79.1	70.4	60.0	48.5	36.2	23.8	12.2
	AT w/ Uniform ϵ			84.4	75.2	63.2	47.8	31.8	16.3	3.6

Condition-level Embedding. Table 7 compares three conditioning strategies: discrete embeddings (`nn.Embedding`), raw continuous scalar input, and continuous Fourier-based embeddings (random Fourier features). Discrete embeddings lack smooth interpolation across noise levels, while raw

continuous inputs provide insufficient expressiveness, and both fail to achieve consistently strong performance across budgets. Fourier-based embeddings resolve this by mapping scalar noise levels into a richer feature space, enabling smoother and more structured conditioning that yields the best overall performance across all perturbation regimes.

Behavior analysis of $\Gamma(\epsilon)$. In CAN, we use the channel-wise conditional modulation $\mathcal{T}(\epsilon)[\mathcal{F}](\mathbf{x}) = \big(1 + \Gamma(\epsilon)\big) \odot \mathcal{F}(\mathbf{x}) + \mathrm{B}(\epsilon)$., where the noise-conditioned parameters $\Gamma(\epsilon)$ and $\mathrm{B}(\epsilon)$ adaptively adjust the embedding h. In particular, $\Gamma(\epsilon)$ controls the rescaling of the embedding. As reported in Table 8, the magnitude of $\Gamma(\epsilon)$ tends to decrease as the noise level ϵ becomes larger. This behavior suggests that under stronger perturbations, CAN applies a stronger modulation to the feature representation. More specifically, the resulting factor $(1 + \Gamma(\epsilon))$ tends to impose a smaller effective scaling on the embedding, which suppresses unstable feature responses and reduces the influence of perturbations in the representation space. This observation may indicate that CAN adaptively contracts the embedding under higher noise levels, helping improve feature stability and robustness.

Table 8: Layer-wise $\Gamma(\epsilon)$ values ($\times 10^{-2}$). Larger noise leads to smaller $\Gamma(\epsilon)$, indicating stronger feature suppression.

Layer ℓ	$\epsilon = \frac{0}{255}$	$\epsilon = \frac{1}{255}$	$\epsilon = \frac{2}{255}$	$\epsilon = \frac{4}{255}$	$\epsilon = \frac{8}{255}$
$\ell = 1$	3.17	2.57	2.08	1.25	-0.05
$\ell = 2$	2.50	1.85	1.31	0.37	-1.10
$\ell = 3$	4.28	3.94	3.61	2.98	1.98
$\ell = 4$	7.07	6.39	5.86	4.97	3.57
$\ell = 5$	2.80	2.66	2.50	2.17	1.64
$\ell = 6$	5.02	4.79	4.55	4.10	3.39
$\ell = 7$	2.00	1.66	1.37	0.87	0.09
$\ell = 8$	3.55	3.05	2.65	2.00	0.97
$\ell = 9$	2.96	2.72	2.50	2.10	1.48
$\ell = 10$	4.48	4.33	4.19	3.96	3.58
$\ell = 11$	1.72	1.37	1.02	0.36	-0.69
$\ell = 12$	3.99	3.61	3.26	2.63	1.63
$\ell = 13$	1.15	1.04	0.93	0.72	0.40
$\ell = 14$	4.65	4.56	4.49	4.36	4.17
$\ell = 15$	0.88	0.73	0.60	0.37	0.02
$\ell = 16$	7.67	7.41	7.22	6.90	6.40

Complexity Analysis. Table 9 compares the cost of supporting 11 perturbation budgets: five predefined budgets and six newly introduced budgets. Maintaining a separate ResNet-18 specialist for each budget requires 2,200 cumulative training epochs, 44.22 GPU hours, and 122.914M stored parameters. CAN instead uses one 200-epoch joint training run followed by six 5-epoch adaptor adaptations, requiring 230 cumulative epochs and 5.52 GPU hours. To support all 11 budgets, it stores one shared backbone, the jointly trained adaptor, and six adapted adaptor copies, for a total of 14.786M parameters. This reduces cumulative epochs by 89.5%, measured GPU time by 87.5%, and stored parameters by 88.0%. At inference, only the adaptor for the selected condition is active: CAN requires 1.1168G FLOPs per query, compared with 1.1158G for a specialist, an increase of 0.09%.

Table 9: Cost of covering 11 perturbation budgets on CIFAR-10 with ResNet-18. Training cost is cumulative; storage retains all 11 conditions. Inference FLOPs are measured per query.

Method	Epochs	GPU h	Params (M)	FLOPs (G)
11 Specialists	2,200	44.22	122.914	1.1158
CAN	230	5.52	14.786	1.1168
Change	-89.5%	-87.5%	-88.0%	$+0.09\%$

4 Related Works

Several lines of work have sought to train a model capable of defending against diverse adversarial threats. One direction exposes the model to multiple threat types during training (Tramer & Boneh, 2019; Maini et al., 2020; Cai et al., 2018), enabling it to handle heterogeneous attacks simultaneously. Another direction redesigns the training objective: methods such as TRADES (Zhang et al., 2019) and MART (Wang et al., 2019) introduce new loss formulations to better balance accuracy and robustness. Scaling model capacity has also shown promise, with larger models exhibiting improved robustness across budgets (Rice et al., 2020). Ensemble-based methods (Pang et al., 2019; Strauss et al., 2017; Cheng et al., 2021) have similarly been explored to improve cross-budget generalization. Despite these efforts, performance gains remain limited across all these approaches: redesigned objectives offer marginal improvement over vanilla AT, while scaling model capacity and ensemble methods both suffer from poor computational scalability, making them impractical for real-world deployment.

5 Conclusion

We study adversarial robustness across diverse condition levels and show that conventional adversarial training generalizes poorly across perturbation regimes due to its budget-specific formulation. We reformulate robustness as a conditional prediction problem and propose CAN, a single model that adapts continuously to the perturbation level by learning a shared representation across adversarial scales. Extensive experiments demonstrate that CAN achieves strong and consistent performance across a wide range of condition levels, generalizes to unseen budgets, and remains parameter-efficient. Our results suggest that modeling adversarial robustness as a continuous conditional learning task provides a more scalable and principled solution for dynamic condition environments.

AI USE STATEMENT

In this work, we used generative AI tools to assist with polishing the writing of the manuscript, including improving clarity, grammar, and phrasing throughout the text. We also used AI coding assistants to help write and debug portions of the experimental code, such as boilerplate training and evaluation scripts and data processing utilities. We did not use generative AI tools to generate research ideas, design experiments, or produce the reported experimental results; the methodology, experimental design, and analysis are the authors' own. All AI-assisted text was reviewed and edited by the authors, and all AI-assisted code was manually reviewed, tested, and verified against the reported results by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies adversarial robustness of machine learning models and does not involve human subjects, personally identifiable information, or the collection of new data. All experiments use publicly available benchmark datasets (e.g., CIFAR-10, WeatherBench, LOL-v2, AG News) under their respective licenses and terms of use. Our method aims to improve the practicality of deploying robust models across a range of operating conditions, and we do not foresee direct negative societal applications beyond those already broadly associated with adversarial robustness research; a more deployable robust model is, if anything, expected to have a net-positive effect on the safety and reliability of downstream systems. We are not aware of any conflicts of interest or sponsorship that influenced this work.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of our results. An anonymized implementation of our method, including training and evaluation code, is available at `https://anonymous.4open.science/r/CAN-CF44/`. The architecture of the proposed Condition-Aware Network (CAN), including the condition embedding and channel-wise conditional modulation, is described in Section 2, and the full two-stage training objective is given in Equations **??** and **??**. Training details, including the optimizer, hyperparameters, and the condition-sampling distribution used in Stage 1 and Stage 2, are provided in Section 3. All datasets used in our experiments are publicly available, with preprocessing and evaluation protocols described in Section 3 and the appendix. Additional experimental results, including the full train-budget $\times$ eval-budget grid, evaluation under multiple attack types, and ablations on the condition embedding and clean-sampling bias, are provided in Appendix B.1 through B.3 to further support reproducibility.

REFERENCES

Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11679–11689. IEEE, 2020.

Qi-Zhi Cai, Min Du, Chang Liu, and Dawn Song. Curriculum adversarial training. *arXiv preprint arXiv:1805.04807*, 2018.

Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European conference on computer vision*, pp. 17–33. Springer, 2022.

Hao Cheng, Kaidi Xu, Chenan Wang, Bhavya Kailkhura, Xue Lin, and Ryan Goldhahn. Mixture of robust experts (more): A robust denoising method towards multiple perturbations. *arXiv preprint arXiv:2104.10586*, 2021.

Hsin-Tien Chiang, Yi-Yen Hsieh, Szu-Wei Fu, Kuo-Hsuan Hung, Yu Tsao, and Shao-Yi Chien. Noise reduction in ecg signals using fully convolutional denoising autoencoders. *Ieee Access*, 7: 60806–60813, 2019.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pp. 79–90, 2023.

Qiyuan Guan, Qianfeng Yang, Xiang Chen, Tianyu Song, Guiyue Jin, and Jiyu Jin. Weatherbench: A real-world benchmark dataset for all-in-one adverse weather image restoration. In *Proceedings of the 33rd ACM international conference on multimedia*, pp. 12607–12613, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*, 2017.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8018–8025, 2020.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pp. 6640–6650. PMLR, 2020.

Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pp. 4970–4979. PMLR, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International conference on machine learning*, pp. 8093–8104. PMLR, 2020.

Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, 126(9):973–992, 2018.

Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, et al. Towards efficient and effective adversarial training. *Advances in Neural Information Processing Systems*, 34:11821–11833, 2021.

Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1709.03423*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International conference on learning representations*, 2019.

Wenhan Yang, Wenjing Wang, Haofeng Huang, Shiqi Wang, and Jiaying Liu. Sparse gradient regularized deep retinex network for robust low-light image enhancement. *IEEE Transactions on Image Processing*, 30:2072–2086, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5728–5739, 2022.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. {PoisonedRAG}: Knowledge corruption attacks to {Retrieval-Augmented} generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3827–3844, 2025.

A Theoretical Analysis

This appendix gives the setting, the concrete architecture the setting instantiates, the proof of Theorem 1, and details on how the block constants Λ_ℓ are obtained.

A.1 Setting

Let $\mathcal{F}_\theta = \mathcal{F}_L \circ \cdots \circ \mathcal{F}_1$ with $\mathcal{F}_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$, where each block collects a linear or convolutional layer together with its inference-mode normalization and activation (see Section A.2 for the concrete architecture this instantiates). The conditioned model is

$$g_\epsilon = \mathcal{T}_L(\epsilon)[\mathcal{F}_L] \circ \cdots \circ \mathcal{T}_1(\epsilon)[\mathcal{F}_1], \qquad \mathcal{T}_\ell(\epsilon)[\mathcal{F}_\ell](\mathbf{h}) = D_\ell(\epsilon)\, \mathcal{F}_\ell(\mathbf{h}) + \mathrm{B}_\ell(\epsilon),$$

where $D_\ell(\epsilon) := \mathrm{diag}\big(1 + \Gamma_\ell(\epsilon)\big)$ is the channel-wise scaling. For a feature map with C_ℓ channels and P_ℓ spatial positions, the flattened operator $D_\ell(\epsilon)$ is diagonal with entries $1 + \gamma_{\ell,c}(\epsilon)$, each repeated P_ℓ times. For fixed ϵ, the pair $(\Gamma_\ell(\epsilon), \mathrm{B}_\ell(\epsilon))$ is a constant, so g_ϵ is a fixed function of the input. Blocks without an adaptor correspond to $\Gamma_\ell \equiv 0$, $\mathrm{B}_\ell \equiv 0$. When a block contains a skip connection, we treat the whole residual unit as one block $\mathcal{F}_\ell$, whose constant satisfies $\Lambda_\ell \le 1 + \Lambda_\ell^{\mathrm{res}}$, where $\Lambda_\ell^{\mathrm{res}}$ is the constant of the residual branch.

A.2 Network Architecture

Figure 6 shows the concrete instantiation of the block decomposition above, used throughout our adversarial robustness experiments: an 18-layer plain convolutional backbone — a VGG-style stack of 3×3 convolutions without residual connections, matching the depth of a standard ResNet-18 while omitting skip connections so that every block $\mathcal{F}_\ell$ can be conditioned independently. The backbone consists of 16 convolutional layers, organized into four stages with channel widths 64, 128, 256, and 512, followed by global average pooling and a final fully connected classification layer.

The condition level ϵ is first mapped to a Fourier feature embedding $\Phi(\epsilon)$ (Section 2), which is then passed through 16 independent lightweight MLPs — the trapezoids in Figure 6, one per convolutional layer — each producing the layer-specific modulation parameters $(\Gamma_\ell(\epsilon), \mathrm{B}_\ell(\epsilon))$ used in the setting above. These parameters are injected into the output of the corresponding layer through the channel-wise affine transformation $\mathcal{T}_\ell(\epsilon)$, allowing every stage of the backbone to be independently rescaled and shifted according to the condition level, rather than conditioning only at a single bottleneck.

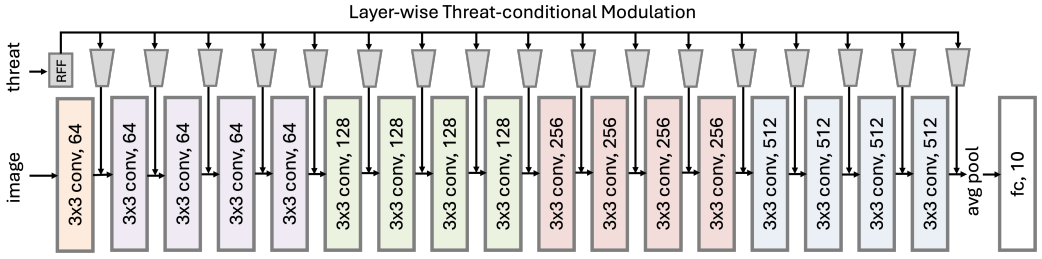


Figure 6: Condition-Aware Network architecture with an 18-layer plain convolutional backbone. Each of the 16 convolutional layers receives its own lightweight adaptor (gray trapezoids), which maps the Fourier-embedded condition level ϵ to a layer-specific channel-wise scale and bias applied to that layer's output.

A.3 Proof of Theorem 1

Proof. **Step 1 (one modulated block).** Fix ϵ and $\mathbf{h}, \mathbf{h}'$ in the domain of block ℓ. Because $\mathrm{B}_\ell(\epsilon)$ does not depend on the input, it cancels in the difference:

$$\begin{aligned}
&\big\|\mathcal{T}_\ell(\epsilon)[\mathcal{F}_\ell](\mathbf{h}) - \mathcal{T}_\ell(\epsilon)[\mathcal{F}_\ell](\mathbf{h}')\big\|_2 \\
&\quad = \big\|D_\ell(\epsilon)\big(\mathcal{F}_\ell(\mathbf{h}) - \mathcal{F}_\ell(\mathbf{h}')\big)\big\|_2 \\
&\quad \le \|D_\ell(\epsilon)\|_{2\to 2}\big\|\mathcal{F}_\ell(\mathbf{h}) - \mathcal{F}_\ell(\mathbf{h}')\big\|_2 \\
&\quad \le \|D_\ell(\epsilon)\|_{2\to 2}\, \Lambda_\ell\, \|\mathbf{h} - \mathbf{h}'\|_2.
\end{aligned}$$

The first inequality follows from the definition of the operator norm, and the second follows from Assumption 1.

Step 2 (norm of the scaling). Since $D_\ell(\epsilon)$ is diagonal, its operator norm is the largest absolute value of its diagonal entries:

$$\|D_\ell(\epsilon)\|_{2\to 2} = \max_c |1 + \gamma_{\ell,c}(\epsilon)| = \|1 + \Gamma_\ell(\epsilon)\|_\infty = s_\ell(\epsilon).$$

Thus, the modulated block is $s_\ell(\epsilon)\Lambda_\ell$-Lipschitz.

Step 3 (composition). The Lipschitz constant of a composition is at most the product of the Lipschitz constants of its components. Hence,

$$\mathrm{Lip}(g_\epsilon) \le \prod_{\ell=1}^{L} s_\ell(\epsilon)\Lambda_\ell = \bar{\Lambda}(\epsilon).$$

For $L = 1$ and $\mathcal{F}_1 = \mathcal{F}_\theta$, this reduces to $\mathrm{Lip}(g_\epsilon) \le \|1 + \Gamma(\epsilon)\|_\infty \Lambda_{\mathcal{F}}$. □

A.4 Computing the Block Constants Λ_ℓ

For a convolutional layer, we bound Λ_ℓ by the spectral norm of the full convolution operator (acting on the whole feature map with the layer's padding and stride), estimated by power iteration, rather than by the spectral norm of the reshaped kernel matrix, which does not bound the operator norm in general. In inference mode, batch normalization is an affine map with per-channel scale $\gamma_c/\sqrt{\sigma_c^2 + \eta}$, and contributes the factor $\max_c |\gamma_c|/\sqrt{\sigma_c^2 + \eta}$. ReLU and GELU-type activations are 1-Lipschitz and contribute a factor at most 1. The resulting Λ_ℓ multiply across layers and are independent of ϵ, since the backbone is shared.

B Additional Experiments

B.1 Full Adversarial Robustness Grid

Table 10: Full train-budget × eval-budget grid on CIFAR10. Rows are AT specialists trained at a fixed budget ϵ_{tr}; columns are evaluation budgets ϵ_{eval}.

Method \ Budget	$\frac{0}{255}$	$\frac{1}{255}$	$\frac{2}{255}$	$\frac{3}{255}$	$\frac{4}{255}$	$\frac{5}{255}$	$\frac{6}{255}$	$\frac{7}{255}$	$\frac{8}{255}$	$\frac{9}{255}$	$\frac{10}{255}$
AT w/ $\epsilon = \frac{0}{255}$	**95.09**	57.86	8.94	1.15	0.14	0.01	0.00	0.00	0.00	0.00	0.00
AT w/ $\epsilon = \frac{1}{255}$	93.09	**86.80**	76.80	64.02	51.01	39.67	31.23	24.93	20.50	17.22	14.48
AT w/ $\epsilon = \frac{2}{255}$	91.11	85.81	**79.28**	70.98	61.39	51.46	42.65	34.67	28.68	23.69	20.16
AT w/ $\epsilon = \frac{3}{255}$	89.82	85.15	79.36	**72.87**	64.92	56.65	48.62	40.40	33.95	28.21	23.40
AT w/ $\epsilon = \frac{4}{255}$	88.51	83.92	78.61	72.45	**65.51**	58.19	50.60	43.32	36.57	30.72	25.89
AT w/ $\epsilon = \frac{5}{255}$	87.21	83.05	78.10	72.55	66.02	**59.04**	52.14	45.58	39.30	33.55	28.33
AT w/ $\epsilon = \frac{6}{255}$	85.85	81.57	76.95	71.66	65.74	59.40	**53.19**	46.68	41.07	35.30	30.13
AT w/ $\epsilon = \frac{7}{255}$	84.90	80.67	76.00	71.20	65.94	59.80	53.66	**47.67**	42.38	36.73	32.02
AT w/ $\epsilon = \frac{8}{255}$	84.07	80.27	75.82	70.99	65.71	60.05	54.47	48.70	**43.10**	38.01	32.88
AT w/ $\epsilon = \frac{9}{255}$	83.02	79.13	75.14	70.41	65.27	60.07	54.80	49.16	43.87	**39.33**	34.54
AT w/ $\epsilon = \frac{10}{255}$	82.85	79.27	74.82	69.72	64.58	59.55	54.63	49.36	44.09	38.78	**34.70**
CAN (Joint Training)	93.85	87.46	80.49	\	66.36	\	\	\	45.16	\	\
CAN (Quick Adaptation)	\	\	\	72.76	\	59.08	53.39	47.89	\	38.95	34.17

To make the one-for-all argument concrete, Table 10 reports the full train-budget × eval-budget grid on CIFAR10. Each row of the specialist block is an adversarially trained model at a single fixed budget ϵ_{tr}, evaluated across all eleven attack budgets from $\frac{0}{255}$ to $\frac{10}{255}$. As expected, every specialist peaks at its own matched budget (bolded on the diagonal) and degrades sharply once evaluated away from it: the $\epsilon = \frac{0}{255}$ specialist scores 95.09% at $\frac{0}{255}$ but collapses to 1.15% by $\frac{3}{255}$, and even the most broadly robust specialist, $\epsilon = \frac{9}{255}$, still loses over 20 points of accuracy moving from its own budget to the clean setting. No single specialist is simultaneously strong across the full spectrum, which is exactly the scalability problem a specialist-per-condition paradigm runs into once the number of budgets of interest grows.

CAN addresses this with a single model trained once, jointly, over a subset of the condition spectrum, $\epsilon \in \{\frac{0}{255}, \frac{1}{255}, \frac{2}{255}, \frac{4}{255}, \frac{8}{255}\}$ (**CAN, Joint Training**). At every one of these budgets, CAN matches or exceeds the corresponding specialist's own best result: 87.46% vs. 86.80% at $\frac{1}{255}$, 80.49% vs. 79.28% at $\frac{2}{255}$, 66.36% vs. 65.51% at $\frac{4}{255}$, and 45.16% vs. 43.10% at $\frac{8}{255}$, while remaining a single set of parameters rather than five independently trained and stored specialists.

We then use the remaining budgets, $\epsilon \in \{\frac{3}{255}, \frac{5}{255}, \frac{6}{255}, \frac{7}{255}, \frac{9}{255}, \frac{10}{255}\}$, to test Stage 2 fast adaptation (**CAN, Quick Adaptation**): for each of these new conditions, we freeze the backbone learned during Joint Training and fine-tune only the lightweight adaptor. Despite never being trained on these budgets during Stage 1 and requiring only a short adaptation step rather than training a new specialist from scratch, CAN comes within one to two points of the corresponding specialist at every new budget, for instance 72.76% vs. 72.87% at $\frac{3}{255}$ and 53.39% vs. 53.19% at $\frac{6}{255}$. Together, these results show that a single CAN can both match dedicated specialists across a predefined condition spectrum and extend to new conditions on demand, at a fraction of the cost of training and storing a specialist for each one.

B.2 Embedding Analysis

To compare transferability across different adversarial levels between vanilla adversarial training and CAN, we quantify the differences in their learned representations under mismatched adversarial conditions. Specifically, we use a ResNet-18 backbone trained on CIFAR-10 and consider five adversarial noise levels:

$$\epsilon_0 = \frac{0}{255}, \epsilon_1 = \frac{1}{255}, \epsilon_2 = \frac{2}{255}, \epsilon_3 = \frac{4}{255}, \epsilon_4 = \frac{8}{255}.$$

For vanilla adversarial training, we independently train five models, one for each adversarial level, resulting in separate parameters $\{\theta_0^*, \theta_1^*, \theta_2^*, \theta_3^*, \theta_4^*\}$. In contrast, CAN requires only a single model θ^* while allowing the adversarial level to be specified through conditioning. We feed clean inputs ($\epsilon_0 = 0$) into all models and extract hidden representations at different depths. This yields two sets of embeddings:

$$\{\mathcal{F}_{\theta_0^*}(\mathbf{x}), \mathcal{F}_{\theta_1^*}(\mathbf{x}), \mathcal{F}_{\theta_2^*}(\mathbf{x}), \mathcal{F}_{\theta_3^*}(\mathbf{x}), \mathcal{F}_{\theta_4^*}(\mathbf{x})\}$$

for vanilla adversarial training, and

$$\{\mathcal{T}(\epsilon_0)[\mathcal{F}_{\theta^*}](\mathbf{x}), \mathcal{T}(\epsilon_1)[\mathcal{F}_{\theta^*}](\mathbf{x}), \mathcal{T}(\epsilon_2)[\mathcal{F}_{\theta^*}](\mathbf{x}), \mathcal{T}(\epsilon_3)[\mathcal{F}_{\theta^*}](\mathbf{x}), \mathcal{T}(\epsilon_4)[\mathcal{F}_{\theta^*}](\mathbf{x})\}$$

for CAN. We consider 7 representations: the input, the output of the input layer, the first convolutional layer, the four residual blocks, and the final logits from the classifier. The results are summarized in Table 11. Overall, the representation differences induced by CAN across adversarial levels are substantially smaller than those produced by vanilla adversarial training. Moreover, in vanilla adversarial training, the independently trained models $\{\theta_i^*\}_{i=0}^4$ produce highly inconsistent embeddings, even when evaluated on the same clean inputs. By contrast, CAN exhibits a gradual increase in representation difference as the condition ϵ_i increases, while the overall discrepancy remains much smaller than that of vanilla adversarial training. These results suggest that the noise-conditioned architecture of CAN learns a sequential family of related mappings, which regularize one another and improve transferability and generalization across adversarial levels.

Table 11: Embedding differences across layers under different adversarial levels. Vanilla adversarial training yields substantially larger representation discrepancies across separately trained models, whereas CAN produces a smoother and more consistent evolution of representations as the condition increases.

Embedding Diff. \ Layer Index	0	1	2	3	4	5	6
$\|\mathcal{F}_{\theta_1^*}(\mathbf{x}) - \mathcal{F}_{\theta_0^*}(\mathbf{x})\|$	0.00	697.79	881.70	600.18	259.36	329.13	52.51
$\|\mathcal{F}_{\theta_2^*}(\mathbf{x}) - \mathcal{F}_{\theta_0^*}(\mathbf{x})\|$	0.00	719.70	898.92	592.84	274.64	349.37	64.08
$\|\mathcal{F}_{\theta_3^*}(\mathbf{x}) - \mathcal{F}_{\theta_0^*}(\mathbf{x})\|$	0.00	716.30	929.93	569.05	284.89	366.81	80.67
$\|\mathcal{F}_{\theta_4^*}(\mathbf{x}) - \mathcal{F}_{\theta_0^*}(\mathbf{x})\|$	0.00	680.91	889.22	540.89	266.21	368.62	98.77
$\|\mathcal{T}(\epsilon_1)[\mathcal{F}_{\theta^*}](\mathbf{x}) - \mathcal{T}(\epsilon_0)[\mathcal{F}_{\theta^*}](\mathbf{x})\|$	0.00	70.41	151.66	82.40	38.74	45.21	17.59
$\|\mathcal{T}(\epsilon_2)[\mathcal{F}_{\theta^*}](\mathbf{x}) - \mathcal{T}(\epsilon_0)[\mathcal{F}_{\theta^*}](\mathbf{x})\|$	0.00	98.56	206.67	118.64	57.27	80.44	29.33
$\|\mathcal{T}(\epsilon_3)[\mathcal{F}_{\theta^*}](\mathbf{x}) - \mathcal{T}(\epsilon_0)[\mathcal{F}_{\theta^*}](\mathbf{x})\|$	0.00	133.29	276.14	167.54	83.72	153.47	53.42
$\|\mathcal{T}(\epsilon_4)[\mathcal{F}_{\theta^*}](\mathbf{x}) - \mathcal{T}(\epsilon_0)[\mathcal{F}_{\theta^*}](\mathbf{x})\|$	0.00	182.45	372.92	233.11	120.40	246.71	86.33

B.3 CLEAN-LEVEL SAMPLING BIAS

We use a clean-biased noise distribution $\mathcal{P} = p \cdot \delta_{\epsilon_0} + (1-p)\,\mathrm{Unif}\{\epsilon_1, \ldots, \epsilon_N\}$, with $p = \frac{M}{M+N}$, where M controls the sampling weight on clean examples. A larger M emphasizes clean training and improves clean accuracy, while a smaller M places more weight on adversarial levels and favors robustness. As shown in Table 12, increasing M generally improves clean accuracy on CIFAR10, but does not consistently improve robustness at larger perturbation budgets. This suggests that a moderate clean bias achieves the best overall trade-off between clean and adversarial performance.

Table 12: Ablation study on the clean-bias coefficient in the training noise distribution. We vary the parameter M in $p = \frac{M}{M+N}$ to control the probability of sampling the clean noise level during training. Larger M increases the clean bias and generally improves clean accuracy, while the best overall trade-off between clean and adversarial performance is achieved at an intermediate bias level.

Backbone	**ResNet18**					**WideResNet-28-10**				
$M : p = \frac{M}{M+N}$	**0/255**	**1/255**	**2/255**	**4/255**	**8/255**	**0/255**	**1/255**	**2/255**	**4/255**	**8/255**
1	92.08	86.44	80.44	66.69	43.45	94.19	88.98	82.62	69.40	48.27
2	92.77	87.01	80.58	66.83	44.85	94.56	88.49	82.32	68.87	46.63
3	93.85	87.46	80.49	66.36	45.16	94.35	88.21	81.06	66.98	45.36
4	93.62	87.11	80.3	66.23	44.23	94.91	87.99	80.76	66.15	43.71

B.4 Image Restoration Visualization

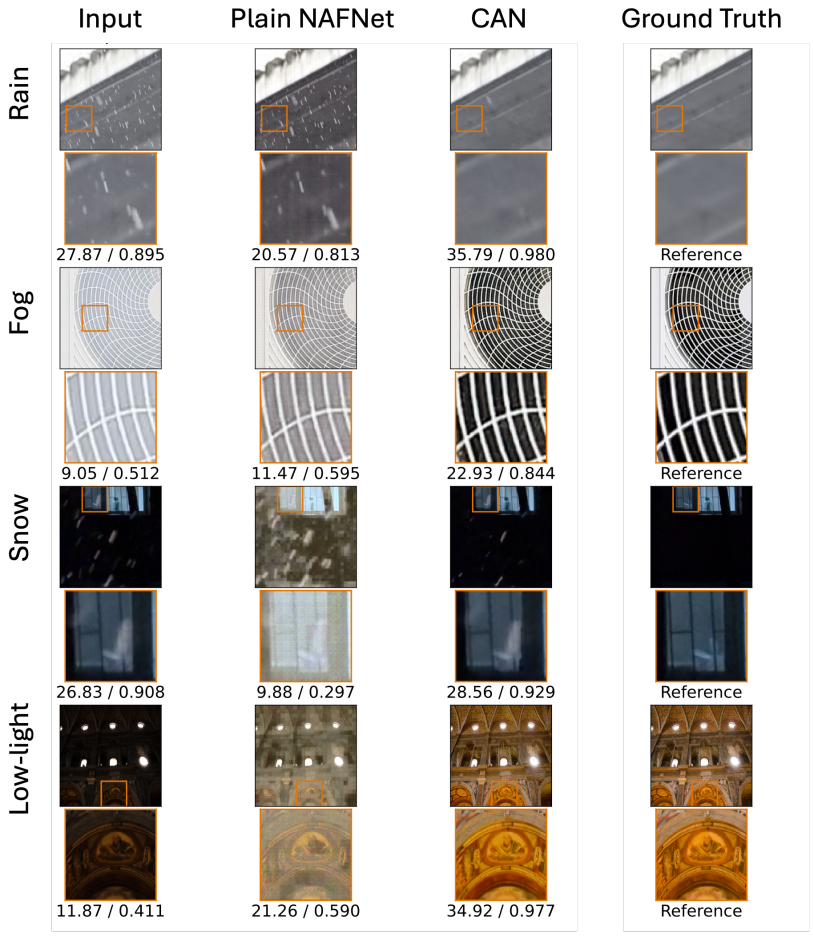


Figure 7: Qualitative restoration results across four degradation conditions: rain, fog, snow, and low-light. For each condition, we show the degraded input, a plain NAFNet-32 specialist, CAN, and the ground-truth reference, with a zoomed crop below each image highlighting fine texture and structure. PSNR / SSIM computed against the reference is reported below each restored image. CAN consistently achieves the highest PSNR/SSIM and best recovers fine structural detail across all four conditions, whereas the fixed NAFNet specialist transfers poorly to conditions it was not trained on, at times reducing PSNR/SSIM below the degraded input itself (e.g., snow: 9.88/0.297 vs. 26.83/0.908).

Figure 7 shows qualitative restoration results across four representative degradation types: rain, fog, snow, and low-light, comparing the degraded input, a plain NAFNet-32 specialist, CAN, and the ground-truth reference, with a zoomed crop below each image highlighting fine texture and structural detail. Each panel reports PSNR / SSIM relative to the reference.

The plain NAFNet specialist, trained under a single fixed condition, transfers poorly once the operating condition changes: on rain and snow it fails to remove the degradation and in fact lowers PSNR/SSIM below the raw, unprocessed input (20.57/0.813 vs. 27.87/0.895 on rain; 9.88/0.297 vs. 26.83/0.908 on snow), effectively corrupting the image further rather than restoring it. On fog and low-light, it offers only a modest improvement over the input (11.47/0.595 and 21.26/0.590, respectively) but still leaves substantial residual degradation, visible in the blurred grille and washed-out fresco detail in the zoomed crops.

CAN, in contrast, consistently achieves great PSNR/SSIM across all four degradation types (35.79/0.980, 22.93/0.844, 28.56/0.929, and 34.92/0.977), closely matching the ground-truth reference both quantitatively and visually, recovering the fine grille structure in the fog example and

the fresco detail in the low-light example that both the raw input and the plain specialist fail to reveal. These qualitative results reinforce the quantitative gains in Table 2: a single condition-aware model generalizes across degradation types where a fixed specialist cannot, and does so without the specialist's risk of actively degrading inputs under a mismatched condition.