FLOAT: Fast Learnable Once-for-All Adversarial Training for Tunable Trade-off between Accuracy and Robustness

Souvik Kundu[1,2]*, Sairam Sundaresan[1], Massoud Pedram[2], Peter A. Beerel[2]
[1]Intel Labs, USA [2]Universiy of Southern California, Los Angeles, USA
{souvikk.kundu, sairam.sundaresan}@intel.com {pedram, pabeerel}@usc.edu

Abstract

Existing models that achieve state-of-the-art (SOTA) performance on both clean and adversarially-perturbed images rely on convolution operations conditioned with feature-wise linear modulation (FiLM) layers. These layers require additional parameters and are hyperparameter sensitive. They significantly increase training time, memory cost, and potential latency which can be costly for resource-limited or real-time applications. In this paper, we present a fast learnable once-for-all adversarial training (FLOAT) algorithm, which instead of the existing FiLM-based conditioning, presents a unique weight conditioned learning that requires ***no*** *additional layer, thereby incurring no significant increase in parameter count, training time, or network latency compared to standard adversarial training. In particular, we add configurable scaled noise to the weight tensors that enables a trade-off between clean and adversarial performance. Extensive experiments show that FLOAT can yield SOTA performance improving both clean and perturbed image classification by up to* $\sim$6% *and* $\sim$10%*, respectively. Moreover, real hardware measurement shows that FLOAT can reduce the training time by up to* $1.43\times$ *with fewer model parameters of up to* $1.47\times$ *on iso-hyperparameter settings compared to the FiLM-based alternatives. Additionally, to further improve memory efficiency we introduce FLOAT sparse (FLOATS), a form of non-iterative model pruning and provide detailed empirical analysis in yielding a three-way accuracy-robustness-complexity trade-off for these new class of pruned conditionally trained models.*

1. Introduction

With the growing usage of DNNs in safety-critical and sensitive applications including autonomous-driving [4] and medical image analysis [11], it has become crucial that they have high classification accuracy on both clean and adversarially-perturbed images [43]. To improve the DNN model performance against these adversarial samples, various defense mechanisms have been proposed including hiding gradients [41], adding noise to parameters [15], and detection of adversaries [32]. In particular, adversarial training [31, 17, 26] has proven to be a consistently effective approach in achieving state-of-the-art robustness. These defenses, however, come at various costs. Firstly, most of these methods suffer from increased training times due to the additional back-propagation overhead caused by generating perturbed images. Secondly, adversarial defenses sometimes cause a significant drop in clean-image accuracy [42], highlighting an accuracy-robustness trade-off that has been explored both theoretically and experimentally [39], [42], [37]. Moreover, the defenses rely on several hyperparameters whose settings force the model to work at a specific point along this trade-off. This is disadvantageous in applications in which the desired trade-off depends on context [43].

A naive solution to this problem is to use multiple models trained with different priorities between clean and adversarial images. This however, comes with the heavy cost of increased training time and inference memory. Alternatively, recent work has proposed training a once-for-all adversarial network (OAT) that supports *conditional learning*

*Part of the work was done when the first author was with USC.

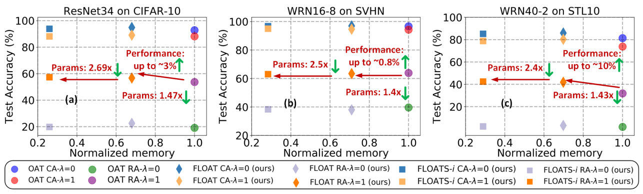

Figure 1. Normalized memory vs. test accuracy for FLOAT and FLOAT with irregular sparsity (FLOATS-i) compared to the existing state-of-the-art OAT for (a) ResNet34, (b) WRN16-8, and (c) WRN40-2, respectively. CA and RA represent clean-image classification accuracy and robust accuracy (accuracy on adversarial images), respectively. For each model we normalized the memory requirement with the maximum memory needed to store corresponding model.

[43], enabling the network to adjust to different input distributions. In particular, after each batch-normalization (BN) layer, they add a feature-wise linear modulation (FiLM) module [35] whose weights are controlled by a parameter λ. For inference, the user sets λ to enable an in-situ trade-off between accuracy and robustness. The disadvantage with this approach is that the added FiLM modules increase the parameter count, training time, and network latency, limiting applicability in resource-constrained, real-time applications. Moreover, our investigation shows that the CA-RA performance of OAT is heavily dependent on the choice of training hyperparameter λs (viz, the accuracy with ResNet34 on CIFAR-10 varies up to 21.97%).

Our contributions. In this paper, our contributions are two-fold. First, in view of the above concerns, we present a *fast learnable once-for-all adversarial training* (FLOAT). In FLOAT, we train a model using a novel mechanism wherein each weight tensor of the model is transformed by conditionally adding a noise tensor based on a binary parameter λ, yielding state-of-the-art (SOTA) test accuracy for clean and adversarial images by in-situ setting $\lambda = 0.0$ and 1.0, respectively. For inference, we further show that model robustness can be correlated to the strength of the noise-tensor scaling factor. This motivates a simple yet effective noise re-scaling approach controlled by an user-provided floating-point parameter that can help the user to have a practical accuracy-robustness trade-off. Because FLOAT does not require additional layers to perform conditioning, it incurs no increase in latency and causes only a negligible increase in parameter count compared to the baseline models. Moreover, compared to OAT, FLOAT training is up to $1.43\times$ faster, attributable to the fact that FLOAT does not require training with intermediate fine-grained values of λs.

Secondly, for efficient deployment of the models to resource-limited edge devices, we present FLOAT sparse (FLOATS), an extension of FLOAT, that not only provides adaptive tuning between RA and CA, but also facilitates high levels of model compression (via pruning) without incurring any additional training time. In particular, we propose and empirically evaluate the efficacy of FLOATS with both irregular and structured channel pruning, namely FLOATS-i and FLOATS-c, respectively. However, despite the potential speed-up on underlying hardware [30], channel pruning often costs classification performance [24, 23] because of its strictly constrained form of sparsity. We thus extend FLOATS to propose a globally-structured locally-irregular hybrid sparsity. In particular, we perform channel reduction through network slimming [48] reducing latency and memory usage, and use irregular pruning in conjunction with this to further reduce memory cost. These new models not only provide compression, but enable an in-situ inference trade-off across accuracy, robustness, and complexity.

To evaluate the merits of FLOAT, we conduct extensive experiments on CIFAR-10, CIFAR-100, Tiny-ImageNet, ImageNet, SVHN, and STL10 with ResNet (on CIFAR, Tiny-ImageNet and ImageNet), WRN16-8, WRN40-2, respectively. As shown in Fig. 1, compared to OAT, FLOAT can provide improved accuracies of up to $\sim 6\%$, and $\sim 10\%$, on clean and perturbed images, respectively, with reduced parameter budgets of up to $1.47\times$. FLOATS can yield even further parameter-efficiency of up to $2.69\times$ with similar CA-RA benefits.

2. Preliminaries

2.1. Notation

Consider a model Φ with L layers parameterized by $\mathbf{\Theta}$ that learns a function $f_\Phi(.)$. For a classification task on dataset X with distribution D, the model parameters $\mathbf{\Theta}$ are learned by minimizing the empirical risk (ERM) as follows

$$\mathcal{L}(f_\Phi(\boldsymbol{x}, \mathbf{\Theta}; t)), \tag{1}$$

where t is the ground-truth class label, $\boldsymbol{x}$ is the vectorized input from $\boldsymbol{X}$, and $\mathcal{L}$ is the cross-entropy loss function.

2.2. Robust Model Training

Several forms of adversarial training (AT) have been proposed to improve robustness [31], [36], [5]. They use clean as well as adversarially-perturbed images to train a model. Projected gradient descent (PGD) attack, recognized as one of the strongest L_∞ adversarial example generation algorithms [31], is typically used to create adversarial images during training. The perturbed image for a PGD-k attack with k as the number of steps is given by

$$\hat{\boldsymbol{x}}^k = \texttt{Proj}_{P_\epsilon(\boldsymbol{x})}(\hat{\boldsymbol{x}}^{k-1} + \sigma \times \texttt{sign}(\nabla_x \mathcal{L}(f_\Phi(\hat{\boldsymbol{x}}^{k-1}, \mathbf{\Theta}; t))) \tag{2}$$

Here, the scalar ϵ corresponds to the perturbation constraint that determines the severity of the perturbation. `Proj` projects the updated adversarial sample onto the projection space $P_\epsilon(\boldsymbol{x})$, which is the ϵ-L_∞ neighbourhood of the benign sample $\boldsymbol{x}$[1]. σ is the attack step-size. For PGD-AT, the model parameters are then learned by the following ERM

$$[\underbrace{(1-\lambda)\mathcal{L}(f_\Phi(\boldsymbol{x}, \mathbf{\Theta}; t))}_{\mathcal{L}_C} + \underbrace{\lambda \mathcal{L}(f_\Phi(\hat{\boldsymbol{x}}, \mathbf{\Theta}; t))}_{\mathcal{L}_A}], \tag{3}$$

where $\mathcal{L}_C$ and $\mathcal{L}_A$ correspond to the clean and adversarial image classification loss components, respectively, weighted by the scalar λ. Hence, for a fixed λ and adversarial strength, the model learns a fixed tradeoff between accuracy and robustness. For example, an AT with λ value of 1

[1]Note that the generated $\hat{\boldsymbol{x}}$ are clipped to a valid range which, for our experiments, is $[0, 1]$.

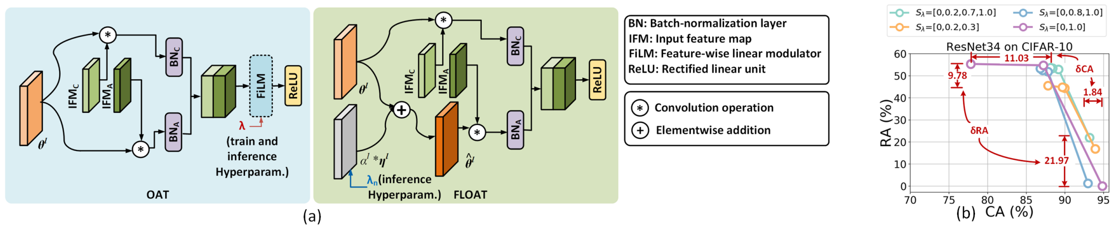

Figure 2. (a) Comparison of a conditional layer between existing FiLM based approach in OAT and proposed approach in FLOAT. (b) Impact of various training λ choices on the conditionally trained OAT. During testing we use $S_\lambda = [0, 0.2, 0.7, 1.0]$.

will allow the model to completely focus on perturbed images, resulting in a significant drop in clean-image classification accuracy. Another strategy to improve model robustness is through the addition of noise to the model weight tensors. For example, [15] introduced the idea of noisy weight tensors with a learnable noise scaling factor and improved robustness against gradient-based attacks. However, this strategy also incurs a significant drop in clean image classification accuracy.

2.3. Conditional Learning

Conditional learning involves training a model with multiple computational paths that can be selectively enabled during inference [44]. For example, [40], [18], [21] enhanced a DNN model with multiple early exit branches at different architectural depths to allow early predictions of various inputs. [48] introduced switchable BNs that enable the network to adjust the channel widths dynamically, providing an in-situ efficient trade-off between complexity and accuracy. Recently, [6] used switchable BNs to support runtime bit-width selection of a mixed-precision network. Another conditional learning approach used feature transformation to modulate intermediate DNN features [19], [47], [10], [43]. In particular, [43] used FiLM [35] to adaptively perform a channel-wise affine transformation after each BN stage that is controlled by the hyperparameter λ of Equation 3. Such conditional training that is able to yield models that can provide SOTA CA-RA trade-off on various λ choices during inference are popularly known as Once-for-all adversarial training (OAT) [43].

Limitations of FiLM-based model conditioning. Each FiLM module in OAT is composed of two fully-connected (FC) layers with leaky ReLU activation functions and dimensions that are integer multiples of the output feature-map channel size. Despite requiring a relatively small number of additional FLOPs, the FiLM module can significantly increase the number of model parameters and associated memory access cost [16]. Moreover, the increased number of layers can significantly increase training time and inference latency [38], thus potentially prohibiting its use in real-time applications.

Additionally, we investigated OAT's performance on the choice of the training λ set (S_λ), as shown in Fig. 2(b). Interestingly, the CA and RA can vary up to 11.03% and 21.97%, respectively. This implies that,*OAT's performance may vary significantly based on both the size and specific values in S_λ*. In particular, the choice of S_λ can significantly impact the robustness at $\lambda = 0$, sometimes leading to no robustness. This implies that to obtain models that yield near optimal CA-RA trade-offs, S_λ must be carefully chosen, implying the need for an additional compute-heavy hyperparameter search or prior user expertise.

3. Proposed Approach

3.1. FLOAT

This section details our FLOAT training strategy. We refer to the conditions for a model being trained on either clean or adversarial images as the two training *boundary conditions*. During training, we use a binary conditioning parameter λ to force the model to focus on either of these two conditions, *removing the need to search a more fine-grained set of λ choices*.

To formalize our approach, consider a L-layer DNN parameterized by $\mathbf{\Theta}$ and let $\boldsymbol{\theta}^l \in \mathbb{R}^{k^l \times k^l \times C_i^l \times C_o^l}$ represent the layer l weight tensor, where C_o^l and C_i^l represent the number of filters and channels per filter, respectively, and k^l represents the kernel height/width. We transform each parameter of $\boldsymbol{\theta}^l$, by adding a noise tensor $\boldsymbol{\eta}^l \in \mathbb{R}^{k^l \times k^l \times C_i^l \times C_o^l}$ scaled by a parameter α^l and conditioned by λ, as follows,

$$\hat{\theta}^l = \theta^l + \lambda \cdot \alpha^l \cdot \eta^l; \quad \eta^l \sim \mathcal{N}(0, (\sigma^l)^2). \tag{4}$$

Note that the standard deviation σ^l of the noise matches that of its weight tensor. $\lambda = 0$ and 1 generate the original weight tensor and its noisy variant, respectively.

As illustrated in Algorithm 1, we train our models by partitioning an image batch $\mathcal{B}$ into two equal sub-batches $\mathcal{B}_1$ and $\mathcal{B}_2$, one with clean (IFM_C) images and the other with perturbed variants (IFM_A) (lines 5 and 7 in Algorithm 1). We use the PGD-7 attack to generate perturbations on the image batch $\mathcal{B}_2$. As illustrated in Fig. 2(a), the original and noisy weight tensors are convolved only with clean and perturbed variants, respectively. Note that the noise scaling factor α^l (line 10) is trainable and its magnitude can be different in each layer to minimize the total training loss. The post-convolution feature maps for clean and adversarial

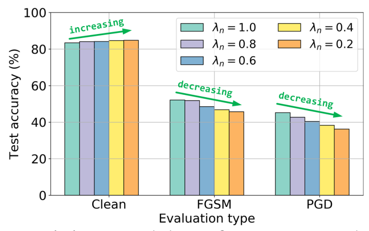

Figure 3. Post-training model performance on both clean and gradient-based attack-generated adversarial images, with different noise re-scaling factor λ_n.

inputs can differ significantly in their respective mean and variances [45], [46]. Therefore, the use of a single BN to learn both distributions may limit the model's performance [43]. To solve this problem, we extend the λ-conditioning to choose between two BNs, BN_C and BN_A, dedicated for IFM_C and IFM_A, respectively.

Our approach differs from previous efforts in several ways. Earlier research performed noise-injection via regularization [3], [28] and perturbed weight tensors [15] to boost model robustness at the cost of a significant accuracy drop on clean images. In contrast, we use noise tensors to transform a shared weight tensor and yield a model that can be configured in-situ to provide SOTA accuracy on either clean or perturbed images. Our approach is similar to λ-conditioning used by [43]. However, instead of transforming activations using added FiLM-based layers trained with multiple values of λ [43], we transform weight tensors using added noise conditioned by binary λ. Compared to [43], we thus require models with significantly fewer parameters and training scenarios, yielding faster training (up to $1.43\times$).

FLOAT generalization with noise re-scaling. One limitation of the FLOAT as proposed above is that it allows the user to choose between two boundary conditions only. This limits applicability when the user is not confident about which condition to use during inference. To motivate more continuous in-situ conditioning, we analyze a ResNet20 model with noisy weight tensors trained with PGD-AT on CIFAR-10 [15]. Post-training, we re-scaled α^l for each layer l, using a new floating-point parameter λ_n to yield $\lambda_n \cdot \alpha^l$. Interestingly, as shown in Fig. 3, as the re-scaling factor decreases, the model robustness decreases and the clean-image accuracy increases.

Based on this observation, we introduce a practical means of post-training in-situ calibration by adding a re-scaling parameter λ_n to the inference model[2]. This allows us to enable a practical accuracy-robustness trade-off in FLOAT during inference. We also define a threshold λ_{th} such that for $\lambda_n > \lambda_{th}$ we select BN_A to perform inference and select BN_C otherwise. [43] selected BN_C and BN_A when $\lambda = 0$ and $\lambda > 0$, respectively. We follow a similar approach by setting $\lambda_{th} = 0$.

[2]Note that λ_n is a continuous variable between 0 and 1 where as λ is binary. $\lambda_n = 0$ and $\lambda_n = 1$ matches the training boundary conditions. OAT, on the other hand, uses a single variable λ that can be any floating point value in $[0, 1]$ during both training and inference.

3.2. FLOAT Extension to Model Compression via Pruning

Pruning is a particular form of model compression that has been effective in reducing model size and compute complexity for large DNNs for resource-constrained deployment [7, 27, 30, 14, 25]. Motivated by these results, we incorporate a form of pruning called sparse learning[3] [27] into FLOAT, which we refer to FLOAT sparse-$irregular$ (FLOATS-i). The resulting approach not only provides a CA-RA trade-off, but also meets a target global parameter density d. In particular, FLOATS ranks every layer based on the normalized momentum of its non-zero parameters. Based on this ranking, FLOATS dynamically allocates more weights to layer that have larger momentum and fewer weights to other layers, while maintaining the global density constraint. To be more precise, let the binary pruning mask be parameterized by the set $\mathbf{\Pi}$ with elements $\boldsymbol{\pi}^l$ representing the mask tensor for layer l. The fraction of 1s in $\boldsymbol{\pi}^l$ is proportional to its relative layer importance evaluated through momentum. During training, the total cardinality of the masked params *always* satisfies the following constraint

$$\sum_{l=1}^{L} \mathtt{card}(\boldsymbol{\theta}^l \odot \boldsymbol{\pi}^l) \leq d \sum_{l=1}^{L} \mathtt{card}(\boldsymbol{\theta}^l). \tag{5}$$

To further ensure that the pruned models have structure and enable speed-up on a wide range of existing hardware [30], we propose FLOATS-c that performs $channel$ pruning. In FLOATS-c, for a layer l, we convert the 4D $\boldsymbol{\theta}^l$ to a 2D weight matrix with C_o^l rows and $(k^l)^2 C_i^l$ columns that is further partitioned in to C_i^l sub-matrices of C_o^l rows and $(k^l)^2$ columns. To evaluate the channel importance, we compute the Frobenius norm (F-norm) of each sub-matrix c by computing $f_c^l = ||\boldsymbol{\theta}^l_{:,c,:,:}||_F^2$. We then keep or remove a channel based on the ranking of f_c^l's, enabling pruning at the channel level. As depicted in Fig. 4(a), the weight heatmaps show that for the same layer FLOATS-c can yield only 20.3% non-zero channels, while FLOATS-i retains all the channels. In fact for the same target d, the channel density can be $10\times$ lower for some layers as compared to that in FLOATS-i. We note that this large scale channel reduction sometimes comes at a non-negligible accuracy drop as shown in Table 1.

A globally structured locally irregular pruning. To simultaneously benefit from aggressive parameter reduction via irregular pruning and width reduction via channel pruning, while maintaining high accuracy, we propose a form of hybrid compression called FLOATS slim. FLOATS slim

[3]Every update of the model happens sparsely, meaning only a fraction of the weights are updated, while others remain as zero.

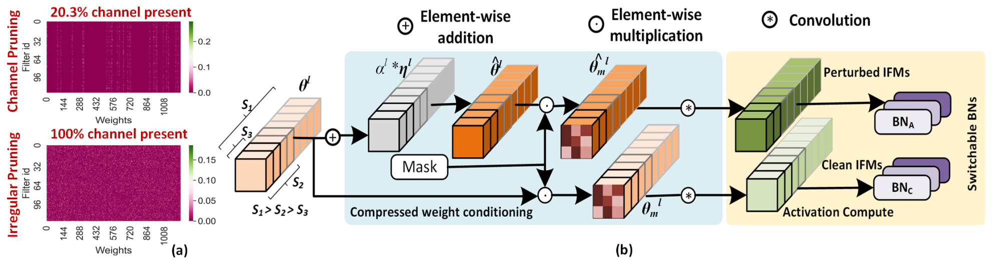

Figure 4. (a) Comparison of channel density (weights plotted in abs. magnitude) for FLOATS irregular and channel, for the 29th CONV layer of WRN40-2 on STL10 while both are trained for $d = 0.3$. (b) Convolutional layer operation path for FLOATS slim. Note, the switchable BNs correspond to BNs for each SF.

Algorithm 1: FLOATS Algorithm

```
Data: Training set $\boldsymbol{X} \sim D$, model parameters $\boldsymbol{\Theta}$, trainable
      noise scaling factor $\boldsymbol{\alpha}$, binary conditioning
      parameter $\lambda$, mini-batch size $\mathcal{B}$, global parameter
      density $d$, initial mask $\boldsymbol{\Pi}$, prune type
      (irregular/channel) $t_p$.
, Output: trained model parameters with density $d$.
$\boldsymbol{\Theta} \leftarrow$ applyMask ($\boldsymbol{\Theta}, \boldsymbol{\Pi}$)
for $i \leftarrow 0$ to to $ep$ do
    for $j \leftarrow 0$ to $n_{\mathcal{B}}$ do
        $\mathcal{B}/2\ (\boldsymbol{X}_{0:\mathcal{B}/2}, \boldsymbol{Y}_{0:\mathcal{B}/2}) \sim D$
        $\mathcal{L}_C \leftarrow$ computeLoss($\boldsymbol{X}_{0:\mathcal{B}/2}, \boldsymbol{\Theta}, \lambda =$
          $0, \boldsymbol{\alpha}; \boldsymbol{Y}_{0:\mathcal{B}/2}$)
        $\hat{\boldsymbol{X}}_{\mathcal{B}/2:\mathcal{B}} \leftarrow$ createAdv($\boldsymbol{X}_{\mathcal{B}/2:\mathcal{B}}, \boldsymbol{Y}_{\mathcal{B}/2:\mathcal{B}}$)
        $\mathcal{L}_A \leftarrow$ computeLoss($\hat{\boldsymbol{X}}_{\mathcal{B}/2:\mathcal{B}}, \boldsymbol{\Theta}, \lambda =$
          $1, \boldsymbol{\alpha}; \boldsymbol{Y}_{\mathcal{B}/2:\mathcal{B}}$)
        $\mathcal{L} \leftarrow 0.5 * \mathcal{L}_C + 0.5 * \mathcal{L}_A$
        updateParam($\boldsymbol{\Theta}, \boldsymbol{\alpha}, \nabla_{\mathcal{L}}, \boldsymbol{\Pi}$)
    end
    updateLayerMomentum ($\boldsymbol{\mu}$)
    pruneRegrow ($\boldsymbol{\Theta}, \boldsymbol{\Pi}, \boldsymbol{\mu}, d$)// Prune fixed % of active and

    // regrow fraction of inactive weights
    $\boldsymbol{\Pi} \leftarrow$ updateMask ($\boldsymbol{\Pi}, t_p, \boldsymbol{\mu}$)
end
```

leverages the idea of slimmable networks [48] to train a model with channel widths that are scaled by a global channel slimming-factor (SF). On top of this, we use FLOATS-i to yield a locally irregular model with even fewer parameters for a specific SF. We perform both of these optimizations simultaneously, training with multiple SFs, including SF $= 1$ (Algorithm detailed in the supplementary material). Note, unlike FLOATS-c, where different layers might have different SFs, FLOATS slim yields uniform SFs for all layers. However, in FLOATS slim, a model with SF< 1.0 is trained as a shared-weight sub-network of the model with SF $= 1.0$, contrasting FLOATS-c, where only one model of a specific d is trained. Fig. 4(b) depicts the weight conditioned convolution operation in FLOATS slim.

4. Experimental Results and Analysis

4.1. Experimental Setup

Models and datasets. To evaluate the efficacy of FLOAT, we performed detailed experiments on six datasets, CIFAR-10, CIFAR-100 [22], Tiny-ImageNet [12], and ImageNet[4] with ResNet [13], SVHN [33] with WRN16-8 [49], and STL10 [8] with WRN40-2 [49].

Hyperparameters and training settings. In order to facilitate a fair comparison, for CIFAR-10, SVHN, and STL10 we used similar hyperparameter settings as [43]. For CIFAR-100, we followed same hyperparameter settings as that with CIFAR-10. For Tiny-ImageNet we trained the model for 120 epochs with an initial learning rate of 0.1 an used cosine decay. Training details on ImageNet is provided in the supplementary. For adversarial image generation during training, we used the PGD-k attack with ϵ and k set to 8/255 and 7, respectively. We initialized the noise scaling-factor α^l for layer l to 0.25 as described in [15]. We used the PyTorch API [34] to implement our models and trained them on a Nvidia GTX Titan XP GPU.

Evaluation metrics. Clean (standard) accuracy (CA): classification accuracy on the original clean test images. Robust Accuracy (RA): classification accuracy on adversarially perturbed images generated from the original test set. We use RA as the measure of robustness of a model. To directly measure the robustness vs accuracy trade-off, we evaluated the clean and robust accuracy values of models generated through FLOAT at various λ values and compared with those yielded through OAT and PGD-AT. We used the average of the best CA and RA values over three different runs with varying random seeds, for each λ value to report in our results.

4.2. Performance of FLOAT

Sampling λ_n. Unless stated otherwise, to evaluate the performance of FLOAT during validation we chose a set of λ_ns as $S_{\lambda_n} = \{0.0, 0.2, 0.7, 1.0\}$. Note that setting λ_n to 0.0 or 1.0 corresponds to the values of λ used during training. Also, we measure the accuracy of FLOAT using

[4]We used part of ImageNet dataset with 50k samples from 100 classes.

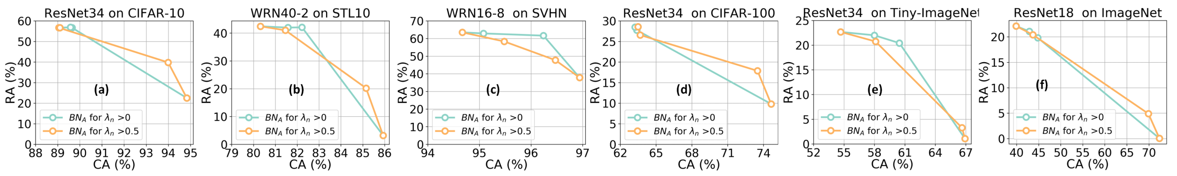
Figure 5. Performance of FLOAT on (a) CIFAR-10, (b) STL10, (c) SVHN, (d) CIFAR-100, (e) Tiny-ImageNet, and (f) ImageNet with various λ_n values sampled from S_{λ_n} for two different λ_{th} for BN_C to BN_A switching. The numbers in the bracket corresponds to (CA, RA) for the boundary conditions of $\lambda = 0$ and $\lambda = 1$. λ_n varies from largest to smallest value from top-left to bottom-right point.

two different settings of λ_{th}, 0.0 (similar to OAT) and 0.5. For $\lambda_{th} = 0.5$, we update the noise scaling factor by using the following simple equation

$$\alpha^l_{new} = \begin{cases} \alpha^l \cdot 2 \cdot \lambda_n; \text{ if } \lambda_n \leq 0.5 \\ \alpha^l \cdot 2 \cdot (\lambda_n - 0.5); \text{ if } 0.5 < \lambda_n \leq 1.0 \end{cases} \tag{6}$$

As depicted in Fig. 5 (a)-(e), the FLOAT models generalize well to yield a semi-continuous accuracy-robustness trade-off. Also, across all the datasets, $\lambda_{th} = 0.5$ yields a more gradual transition between the two boundary conditions. Consider the setting where $\lambda_n = 0.2$. With $\lambda_{th} = 0.5$, we observe a 4.76% improvement in CA and a reduction in RA of 15.9% on average over all five datasets when compared with $\lambda_{th} = 0.0$. The improvement in clean accuracy here can be attributed to the use of BN_C. However, this configuration shows a drop in CA and an improvement in RA when compared to the configuration where $\lambda_n = 0.0$. This can be attributed to the use of noisy weights (refer to Eq. 6) during inference. Thus, it can be concluded that a user who cares more about clean image performance than adversarial robustness, should set $\lambda_{th} > 0.0$ to see a less abrupt drop in CA. Note that, because the generation of adversarial images is noisy, it is not always true that increasing λ will always significantly improve robustness. Consequently, in some cases, we obtain improved clean image performance without a significant drop in robustness.

4.3. Comparison with OAT and PGD-AT

We trained the benchmark models following OAT and PGD-AT with λs sampled from a set $S_\lambda = S_{\lambda_n}$ on three datasets, CIFAR-10, SVHN, and STL10.

Discussion on CA-RA trade-off. Fig. 6(a)-(c) show the comparison of FLOAT with OAT and PGD-AT in terms of CA-RA trade-offs. The FLOAT models show similar or superior performance at the boundary conditions as well as at intermediate sampled values of λ. In particular, compared to OAT and PGD-AT models, FLOAT models can provide an improved RA of up to 14.5% (STL10, $\lambda = 0.2$) and 22.52% (CIFAR-10, $\lambda = 0.0$), respectively. FLOAT also provides improved CA of up to 6.5% (STL10, $\lambda = 1.0$) and 6.96% (STL10, $\lambda = 1.0$), compared to OAT and PGD-AT generated models, respectively. Interestingly, for both FLOAT and OAT, in all the plots we generally see a sharp

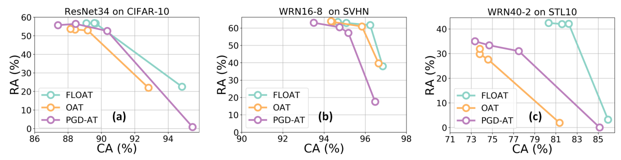
Figure 6. Performance comparison of FLOAT with OAT and PGD-AT generated models on (a) CIFAR10, (b) SVHN, and (c) STL10. λ varies from largest to smallest value in S_λ for the points from top-left to bottom-right.

drop in robustness while moving from top-left to bottom-right. This can be attributed to to the switch from BN_A to BN_C based on the λ_{th}, in the forward pass of the inference model.

Discussion on training time and inference latency. Due to the presence of the additional FiLM modules, OAT requires more time than standard PGD-AT to train. However, a single PGD-AT training can only provide a fixed accuracy-robustness trade-off. For example, to have trade-off with 4 different λs PGD-AT training time increases proportionally by a factor of 4. FLOAT, on the contrary, due to absence of additional layers, trains faster than OAT. In particular, Fig. 7(a) shows the normalized per-epoch training time (averaged over 200 epochs) of OAT and PGD-AT are, respectively, up to $1.43\times$ and $1.37\times$ slower than FLOAT.

Network latency increases with the increase in the number of layers for both standard and mobile GPUs [29], [38], primarily because layers are operated on sequentially [38]. The additional FiLM modules in OAT significantly increase the layer count. For example, for each bottleneck layer in ResNet34, OAT requires two FiLM modules, yielding a total of four additional FCs per bottleneck. On the other hand, FLOAT, similar to a single PGD-AT trained model, requires no additional layers or associated latency, making it more attractive for real-time applications.

Discussion on model parameter storage cost. Unlike OAT, where the FiLM layer FCs significantly increase the parameter count, the additional BN layers and scaling factors of FLOAT represent a negligible increase in parameter count. In particular, assuming parameters are represented with 8-bits, a FLOAT ResNet34 has only 21.28 MB memory cost compared to 31.4MB for OAT. Fig. 7(b) shows that FLOAT models, similar to PGD-AT:1T, can yield up to $1.47\times$ lower memory.

Discussion on FLOPs. Compared to the standard PGD-AT, FLOAT incurs additional compute cost of addition of

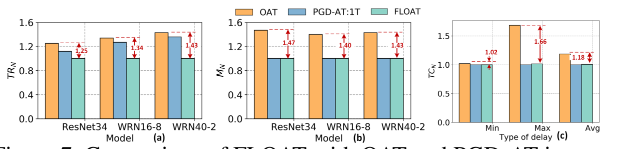
Figure 7. Comparison of FLOAT with OAT and PGD-AT in terms of (a) normalized training time per epoch and (b) model parameter storage (neglecting the storage cost for the BN and $\boldsymbol{\alpha}$) (c) CONV layer compute delay on conventional ASIC (using the delay model of Eq. 7, 8, and 9) architecture [1] evaluated on ResNet34 for CIFAR-10. Note, PGD-AT:1T yields 1 model for each λ choice.

noise with the weight tensor during forward pass. For example, for ResNet34 with ∼21.28 M parameters, FLOAT needs similar number of additions for noisy weight transformation. However, compared to the total operations of ∼1.165 GFLOPs, the transformation adds on 1.182% additional computation. Moreover, as a single addition can be up to 32× cheaper than a single FLOP [16], we can gracefully ignore such transformation cost in terms of FLOPs. OAT, on the other hand, also incurs negligibly less FLOPs overhead of up to only ∼1.7% [43].

Discussion on compute delay in Von-Neumann ASIC hardware. A neural network deployed on a conventional Von-Neumann hardware has two dominant operation types: memory read and Multiply-accumulate (MAC). Based on the assumption that these operations are sequential, as in [1, 20], the convolution layer delay to compute C_o^l output-features can be estimated

$$\tau_{conv} \approx \lceil \frac{(k^l)^2 C_i^l C_o^l}{(B_{IO}/B_W) N_{bank}} \rceil \times \tau_{read} + \lceil \frac{(k^l)^2 C_i^l C_o^l}{N_{Mult}} \rceil \times H_o^l W_o^l \tau_{mult}. \quad (7)$$

where B_{IO} is the memory input-output (IO) bandwidth and B_W is the bit-width of each weight stored in memory. N_{bank} and N_{mult} corresponds to the number of hardware memory banks and multiply units. Similar to earlier literature [1, 20], for a standard hardware we assume the values of B_{IO}, B_W, N_{bank}, and N_{mult} to be 16, 8, 4, and 175, respectively. A single memory read and multiply operation time is denoted by τ_{read} and τ_{mult}, respectively. Their values for a 65nm CMOS process technology are $9ns$ and $4ns$, respectively [20]. Based on similar assumptions, the delay model for modified CONV layer l for FLOAT (τ_{conv}^F) and OAT (τ_{conv}^O) can be estimated as

$$\tau_{conv}^F \approx \lceil \frac{(k^l)^2 C_i^l C_o^l}{(B_{IO}/B_W) N_{bank}} \rceil \times \tau_{read} + \lceil \frac{(k^l)^2 C_i^l C_o^l}{N_{Mult}} \rceil \times (1 + H_o^l W_o^l) \tau_{mult}, \quad (8)$$

$$\tau_{conv}^O \approx \lceil \frac{(k^l)^2 C_i^l C_o^l + 2C_o^l + 4(C_o^l)^2}{(B_{IO}/B_W) N_{bank}} \rceil \times \tau_{read} + \lceil \frac{(k^l)^2 C_i^l C_o^l}{N_{Mult}} \rceil \times H_o^l W_o^l \tau_{mult} + \lceil \frac{2C_o^l + 4(C_o^l)^2}{N_{Mult}} \rceil \times \tau_{mult}. \quad (9)$$

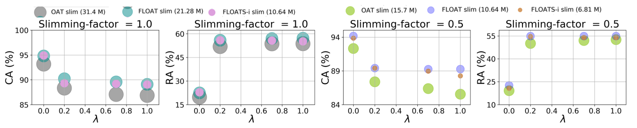
Figure 8. Performance comparison of FLOAT slim, FLOATS($-i$) slim with OAT slim. We used ResNet34 on CIFAR-10 to evaluate the performance.

Here, the first term corresponds to the read delay and remaining term(s) correspond to the delay associated with the multiplications. We ignore the energy associated with reading α^l because it is negligible compared to the read energy for the other model parameters. Based on these Eqs, Fig. 7(c) shows the minimum, maximum, and average normalized delays with respect to the τ_{conv}. In particular, conditional CONV layer delay of FLOAT can be up to 1.66× faster compared to that of OAT, illustrating its efficacy on conventional architecture.

4.4. Performance of FLOATS

Table 1 shows the performance of FLOATS with irregular, channel, and slimmable compression. The FLOATS slim model was trained with two representative SFs of 1.0 and 0.5 with a global target density $d = 0.3$. We report its performance with SF$= 0.5$. Here, *compression ratio* (CR) and *channel reduction factor* (CRF) are computed as $\frac{1}{d}$ and $\frac{100}{\text{\% of total channels present}}$, respectively. Compared to FLOATS-c, FLOATS slim requires 1.62× less storage, results in up to 2× speed-up, and yields 2.24% higher classification accuracy. Moreover, FLOATS slim provides us with a unique three-way trade-off between robustness, accuracy, and complexity, requiring only single training pass.

Fig. 8 illustrates the efficacy of FLOAT slim compared to OAT slim. FLOAT slim provides significantly improved performance for all tested values of λ for both the SFs. In particular, FLOAT slim yields up to 3.6% higher accuracy. Adding sparsity, FLOATS slim yields similar accuracy improvement with up to 2.95× less parameters. Moreover, GPU hardware measurements show that our slimmable networks trains up to 1.90× faster compared to OAT slim.

4.5. Generalization and Gradient Obfuscation Tests

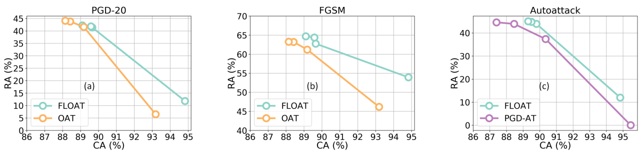
Figure 9. Performance comparison of FLOAT with OAT on (a) PGD-20 and (b) FGSM attack generated images. (c) CA-RA plot of FLOAT vs. PGD-AT on autoattack. All evaluations are done with ResNet34 on CIFAR-10. λ varies from largest to smallest value in S_λ for the points from top-left to bottom-right.

To demonstrate the generalization of FLOAT models on different attacks, we show their performance on im-

Algorithm	Acc. % ($\lambda = 0.0$)		Acc. % ($\lambda = 1.0$)		CR ↑	CRF ↑	Reduced storage	Potential speed-up
	CA	RA	CA	RA				
FLOAT	**94.83**	**22.52**	**89.1**	**56.71**	$1\times$	$1\times$	✗	✗
FLOATS-i	94.12	18.7	88.6	55.92	**10**$\times$	$1\times$	✓✓	✗
FLOATS-c	93.84	17.2	86.87	53.2	$2.94\times$	$1.54\times$	✓	✓
FLOATS slim	94.26	19.1	88.9	55.44	$4.76\times$	**2**$\times$	✓✓	✓

Table 1. Performance comparison between different compressed FLOAT variants trained on CIFAR-10 with ResNet34. ✓✓, ✓, and ✗ indicate aggressive, non-aggressive, and no reduction, respectively, compared to the baseline of FLOAT.

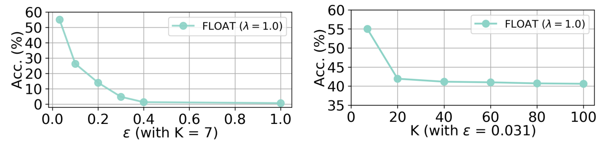

Figure 10. PGD attack performance as a function of (a) bound ϵ and (b) attack iterations K for $\lambda = 1.0$. We observe a similar trend for other inference λs.

Checks to identify gradient obfuscation	Pass
i) Single-step attack performs better compared to iterative attacks	✓
ii) Black-box attacks performs better compared to white-box attacks	✓
iii) Increasing perturbation bound can't increase attack strength	✓
iv) Unbounded attacks can't reach $\sim$100% success	✓
v) Adversarial example can be found through random sampling	✓

Table 2. Checklist tests for characteristic behaviors caused by obfuscated and masked gradients [2].

ages adversarially-perturbed through PGD-20 and FGSM attacks. We follow [43] to generate the PGD-20 perturbations and set the number of steps to 20, keeping other hyperparameters the same as PGD-7. For FGSM, we make $\epsilon = 8/255$ following [43]. As shown in Fig. 9(a)-(b), under both the attacks, FLOAT can achieve in-situ accuracy-robustness trade-offs similar to that of OAT. Moreover, we have analyzed FLOAT's robustness with an ensemble of parameter-free attacks, namely the 'random' variant of autoattack [9][5]. Details of the autoattack hyperparameters are provided in the Supplementary Materials. As depicted in 9(c), compared to the PGD-AT yielded models, FLOAT consistently yields better RA with similar or improved CA.

We now show results on the checklist to examine whether the robustness improvement is caused by obfuscated and masked gradient (see Table 2) as proposed in [2]. Fig. 9(a) and (b) show, single-step FGSM attack performs poorly compared to the iterative counter-part PGD. This certifies the success of Test (i), as listed in Table 2. To perform Test (ii), we trained two ResNet34 models (A and B) on CIFAR-10 and used A to generate black-box (BB) adversarial images for B. The test passes because the model robustness improves on BB attack generated images by $\sim$13%($\lambda = 1.0$), compared to that on white-box (WB) adversaries (Fig. 5(a)), indicating weaker BB than WB attacks. To verify Tests (iii) and (iv) we analyzed ResNet34 on CIFAR-10 with increasing attack bound ϵ. As shown in Fig. 10(a), the classification accuracy decreases as we increase ϵ and finally reaches an accuracy of $\sim$0%. Test (v) can fail only if gradient based attacks cannot provide adversarial inputs for the model to misclassify. However, despite adv training, both FGSM and PGD, the gradient attack variants, can fool the network despite our training.

We further evaluated the model with increasing attack strength by increasing the PGD iteration count K. As Fig. 10(b) shows, after an initial drop in robustness, the model robustness nears an asymptote. In contrast, if the success of FLOAT models arose from the incorrect gradient of a single sample, increasing the attack iterations would have broken the defense completely [2]. Thus both BPDA and Autoattack checklist validate the "real" robustness improvement.

[5]We have followed the official repo https://github.com/fra31/auto-attack to generate the attack.

5. Conclusions

This paper addresses the largely unexplored problem of enabling an in-situ inference trade-off between accuracy, robustness, and complexity. We present FLOAT, a fast learnable once-for-all adversarial training that uses model conditioning to capture the different feature-map distributions corresponding to clean and adversarial images. FLOAT transforms its weights using conditionally added scaled noise and dual batch normalization structures to distinguish between clean and adversarial images. The approach avoids increasing the layer count, unlike other state of the art alternatives, and thus does not suffer from increased network latency. FLOAT models can be configured in-situ to dynamically adjust the model's accuracy, robustness, and complexity based on the image scenario without requiring any additional trained weights, making them suitable for portable AI-enabled devices having stringent storage and energy-budget limitations. We then extend FLOAT to include sparsity to further reduced complexity and latency providing an in-situ trade-off including model complexity. We consider auto selection of the λs based on prediction confidence as an interesting future research direction.

Potential negative societal impact. Adversarial training compute cost and time is $\sim 7\times$ costlier than standard DNN training. This can be further translated to significantly higher carbon footprint. As opposed to standard training, despite FLOAT's significantly high once-for-all training cost, we believe its ability to generate in-situ configurable models to adjust accuracy and robustness trade-off can dramatically reduce the iterative training costs.

References

[1] Mustafa Ali, Akhilesh Jaiswal, Sangamesh Kodge, Amogh Agrawal, Indranil Chakraborty, and Kaushik Roy. Imac: In-memory multi-bit multiplication and accumulation in 6t sram array. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(8):2521–2531, 2020.

[2] Anish Athalye et al. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[3] Alberto Bietti, Grégoire Mialon, and Julien Mairal. On regularization and robustness of deep neural networks. 2018.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[5] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[6] Adrian Bulat and Georgios Tzimiropoulos. Bit-mixer: Mixed-precision networks with runtime bit-width selection. *arXiv preprint arXiv:2103.17267*, 2021.

[7] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[9] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[10] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. *arXiv preprint arXiv:1707.00683*, 2017.

[11] Tianyu Han, Sven Nebelung, Federico Pedersoli, Markus Zimmermann, Maximilian Schulze-Hagen, Michael Ho, Christoph Haarburger, Fabian Kiessling, Christiane Kuhl, Volkmar Schulz, et al. Advancing diagnostic performance and clinical usability of neural networks via adversarial training and dual batch normalization. *Nature Communications*, 12(1):1–11, 2021.

[12] Lucas Hansen. Tiny ImageNet challenge submission. *CS 231N*, 2015.

[13] 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*, pages 770–778, 2016.

[14] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

[15] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–597, 2019.

[16] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[17] Weizhe Hua, Yichi Zhang, Chuan Guo, Zhiru Zhang, and G Edward Suh. Bullettrain: Accelerating robust neural network training via boundary example mining. *Advances in Neural Information Processing Systems*, 34, 2021.

[18] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[19] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[20] Mingu Kang, Sungmin Lim, Sujan Gonugondla, and Naresh R Shanbhag. An in-memory vlsi architecture for convolutional neural networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(3):494–505, 2018.

[21] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning*, pages 3301–3310. PMLR, 2019.

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

[23] Souvik Kundu, Yao Fu, Bill Ye, Peter A Beerel, and Massoud Pedram. Towards adversary aware non-iterative model pruning through d ynamic n etwork r ewiring of dnns. *ACM Transactions on Embedded Computing Systems (TECS)*, 2022.

[24] Souvik Kundu, Mahdi Nazemi, Peter A Beerel, and Massoud Pedram. Dnr: A tunable robust pruning framework through dynamic network rewiring of dnns. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 344–350, 2021.

[25] Souvik Kundu, Mahdi Nazemi, Massoud Pedram, Keith M Chugg, and Peter A Beerel. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Transactions on Computers*, 69(7):1045–1058, 2020.

[26] Souvik Kundu, Massoud Pedram, and Peter A Beerel. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5209–5218, 2021.

[27] Souvik Kundu and Sairam Sundaresan. Attentionlite: Towards efficient self-attention models for vision. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2225–2229. IEEE, 2021.

[28] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[29] Zhengang Li, Geng Yuan, Wei Niu, Pu Zhao, Yanyu Li, Yuxuan Cai, Xuan Shen, Zheng Zhan, Zhenglun Kong, Qing Jin, et al. Npas: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14255–14266, 2021.

[30] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[31] 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.

[32] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147, 2017.

[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[35] 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.

[36] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[37] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. *arXiv preprint arXiv:1804.11285*, 2018.

[38] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P Namboodiri. Hetconv: Heterogeneous kernel-based convolutions for deep cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4835–4844, 2019.

[39] Ke Sun, Zhanxing Zhu, and Zhouchen Lin. Towards understanding adversarial examples systematically: Exploring data size, task and model factors. *arXiv preprint arXiv:1902.11019*, 2019.

[40] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[41] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[42] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[43] Haotao Wang, Tianlong Chen, Shupeng Gui, Ting-Kuei Hu, Ji Liu, and Zhangyang Wang. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. *arXiv preprint arXiv:2010.11828*, 2020.

[44] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[45] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

[46] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. *arXiv preprint arXiv:1906.03787*, 2019.

[47] Shuai Yang, Zhangyang Wang, Zhaowen Wang, Ning Xu, Jiaying Liu, and Zongming Guo. Controllable artistic text style transfer via shape-matching gan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4442–4451, 2019.

[48] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

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