GradAuto: Energy-oriented Attack on Dynamic Neural Networks

Jianhong Pan[1‡], Qichen Zheng[1‡], Zhipeng Fan[2], Hossein Rahmani[3], Qiuhong Ke[4], and Jun Liu[1⋆]

[1] Information Systems Technology and Design, Singapore University of Technology and Design, Singapore
{jianhong_pan,qichen_zheng,jun_liu}@sutd.edu.sg
[2] Tandon School of Engineering, New York University, Brooklyn NY, USA
zf606@nyu.edu
[3] School of Computing and Communications, Lancaster University, United Kingdom
h.rahmani@lancaster.ac.uk
[4] Department of Data Science & AI, Monash University, Australia
qiuhong.ke@monash.edu

Abstract. Dynamic neural networks could adapt their structures or parameters based on different inputs. By reducing the computation redundancy for certain samples, it can greatly improve the computational efficiency without compromising the accuracy. In this paper, we investigate the robustness of dynamic neural networks against energy-oriented attacks. We present a novel algorithm, named GradAuto, to attack both dynamic depth and dynamic width models, where dynamic depth networks reduce redundant computation by skipping some intermediate layers while dynamic width networks adaptively activate a subset of neurons in each layer. Our GradAuto carefully adjusts the direction and the magnitude of the gradients to efficiently find an almost imperceptible perturbation for each input, which will activate more computation units during inference. In this way, GradAuto effectively boosts the computational cost of models with dynamic architectures. Compared to previous energy-oriented attack techniques, GradAuto obtains the state-of-the-art result and recovers 100% dynamic network reduced FLOPs on average for both dynamic depth and dynamic width models. Furthermore, we demonstrate that GradAuto offers us great control over the attacking process and could serve as one of the keys to unlock the potential of the energy-oriented attack. Please visit `https://github.com/JianhongPan/GradAuto` for code.

1 Introduction

Deep neural networks(DNNs) have made great progress in a large variety of computer vision tasks such as image classification [12, 22, 26, 36, 37], segmentation [1, 8, 21, 38, 39] and object detection [5, 16, 21, 33, 35]. However, with the

⋆ Corresponding author.
‡ Both authors contributed equally to this research.

advance of the research on neural architectures as well as the developments in hardware, DNNs have become deeper and deeper with millions or even billions of parameters nowadays, especially after the introduction of the Transformer [43]. The computation heavy models pose great threats to the deployment to embedded and mobile devices and motivate more researches on developing more efficient models [9, 10] to accelerate the training/inference.

Dynamic Neural Network [19] renders one potential path towards acceleration in inference stage by adaptively executing a subset of layers/neurons conditioned on the properties of the input sample. Typically, gating mechanism is introduced into dynamically neural networks to adaptively determine if their corresponding layers/neurons should be executed based on the input. By skipping the redundant computations within the models on the fly, dynamic neural networks provide a better trade-off between the accuracy and the efficiency. Moreover, the adaptive mechanism enlarges the parameter space and unlocks more representation power with better interpretability [19].

Despite the fast developments in dynamic neural networks architecture and training techniques, the robustness of dynamic neural network to adversarial examples have only attracted attention [20] recently. Studying the adversarial attacks on the dynamic neural network provides an alternative view over the robustness on the efficiency of the model. As shown in ILFO [20], although dynamic neural networks offer notable advantages on accuracy, computational efficiency, and adaptiveness, these networks are highly vulnerable to energy-oriented attacks, i.e. adversarial attacks aimed at boosting the energy consumption and computation complexity.

By adding a well-designed perturbation on the input, the adversarial examples can easily activate more gates controlling the execution of the modules, therefore incuring more computations and drastically reducing the FLOPs (floating-point operations per second) saved by the state-of-the-art dynamic networks. For example, with the adversarial attack technique proposed in ILFO [20], the computation overhead of the state-of-the-art SkipNet [45] recovered by more than 84% and 81% on CIFAR-10 and ImageNet, respectively. ILFO [20] establishes a good baseline for energy-oriented adversarial attacks on dynamic depth neural networks. Nevertheless, ILFO is not evaluated on dynamic width networks. In our experiments, we adapt ILFO to attack dynamic width networks.

Furthermore, we identify two more key pitfalls of the existing energy-oriented attacks towards dynamic neural networks like ILFO, as shown in Sec. 4.1 and Sec. 4.2. First, the drastic difference between the magnitude of the gradients across the gates could potentially lead to some gates dominate the others during the training process. This imbalance between the gradient magnitude across different gates will impede the training and convergence of the model. Second, the gradients of different gates will not always in harmony. Gradients computed from inactivated gates could potentially disagree with the one that computed from activated gates, which will deactivate the already activated gates after a few updates.

To mitigate such issues at their root, we propose the GradAuto, a unified method for energy-oriented attacks on both dynamic depth and width networks. Specifically, we directly modify the gradient magnitudes to meet Lipschitz continuity, which will provide convergence guarantee during the training phase. To address the second pitfall, we make the gradient direction of the deactivated gate orthogonal to the gradient direction of the activated gates in every update step, which mitigates the influence of the gates that have already been optimized.

In summary, we propose GradAuto to perform effective energy-oriented attacks against the dynamic neural networks. The contribution of our work is three folds: (1) We provide a unified formulation to construct adversarial samples to attack both the dynamic depth and width networks. (2) To address the drastic magnitude differences among gradients as well as the disagreement in gradients for activated/inactivated gates, we propose GradAuto to rectify both of them. (3) We demonstrate the efficacy of our algorithm on multiple dynamic neural network structures as well as various datasets. Our GradAuto bumps up more computations with less perceptable perturbations.

2 Related Work

2.1 Dynamic Neural Networks

Dynamic neural network [31, 44, 45, 48] addresses the trade-off between the accuracy and the efficiency of deep neural networks by adaptively adjusting model architectures to allocate appropriate computation conditioned on each instance. As a result, the redundant computations on those "easy" samples are reduced and the inference efficiency is improved [19]. Below we briefly review the dynamic depth networks and dynamic width networks. A more comprehensive review of dynamic neural networks can also be referred to [19].

Dynamic depth networks achieve efficient inference in two ways, early termination and conditional skipping. The early-termination based models could optionally finish the inference early from shallower layers, when high confidences on the predictions have been achieved. The conditional-skipping based models skip the execution of a few intermediate layers/blocks, while the prediction is always obtained at the end of the model. Adaptive Computation Time (ACT) [17] augments an RNN with a scalar named halting score to save computational cost. Figurnov et al. [14] further extended this idea to ResNet for vision task by applying ACT to each groups of residual blocks. SkipNet [45] attempts to skip convolutional blocks using an RL-based gating network.

Dynamic width networks selectively activate multiple components of the same layer, such as channels and neurons based on each instance. Early studies [3, 4, 11] achieve dynamic width by adaptively controlling the activation of neurons. The MoE [13, 27] structure builds multiple "experts" (which can be complete models or network modules) in a parallel structure, and dynamically weights the output of these "experts" to get the final prediction. Dynamic channel pruning in CNNs [25, 32, 34, 46, 47] adaptively activates different convolution channels based

on different instances, thereby achieving computational efficiency improvements while maintaining model capacity.

In this paper, we propose a unified algorithm to generate adversarial samples to perform energy-oriented attacks. We showcase the efficacy of our algorithm on both the dynamic depth network (SkipNet [45] and SACT [14]) as well as dynamic width network (ManiDP [42]). Thanks to the general formulation, our algorithm could be extended to other instances of dynamic depth and width networks easily.

2.2 Energy-oriented Attack

Although dynamic neural network is appealing in reducing the computation overhead specific to the input samples, its robustness under various adversarial attacks remains unclear. The adversarial attacks could be broadly separated into accuracy-oriented attacks and energy-oriented attacks. In accuracy-oriented attacks, the adversary aims at altering the predictions of the target models while in energy-oriented attacks, the adversary focuses on delaying the inference speed of the target model by incurring more computations within the model. With the extra computation overhead, the energy consumption of inference also increases accordingly, leaving the dynamic neural network no longer efficient and therefore defeats the whole purpose of the dynamic neural networks.

Important as it is, the study on adversarial attacks on dynamic neural networks is quite limited. ILFO [20] was the first work investigating the robustness of dynamic neural networks. Specifically, they study the energy-oriented adversarial attacks on dynamic depth networks and leverage the intermediate output within the dynamic neural network to infer the energy consumption of each layer. Deepsloth [23] attacks the early-termination based dynamic neural nets. An adversarial example crafting technique is proposed to slowdown the inference speed.

Different from existing methods that mainly focus on studying the robustness of the dynamic depth network, we provide a more universal formulation that could adapt to the attack on both the dynamic depth network and dynamic width network. Moreover, we also showcase that directly searching for the adversarial examples based on gradients could be unstable and sub-optimal. To address this, we additionally propose two remedies to rectify the gradients, leading to improved adversary performance with less perceptible perturbations.

3 Computational Complexity Attack

3.1 Introduction of Two Types of Dynamic Neural Architectures

Depth-Dynamic Neural Architecture, such as SkipNet [45], adjusts the network depth by skipping some of the network layer to reduce the computational complexity. Given a conditional skipping network, the output feature maps $\boldsymbol{x}_{l+1} \in \mathbb{R}^{C_{l+1} \times H_{l+1} \times W_{l+1}}$ from the gated layer $l+1$ can be computed by

$$\boldsymbol{x}_{l+1} = g_l \cdot F_l(\boldsymbol{x}_l) + (1 - g_l) \cdot \boldsymbol{x}_l; \tag{1}$$

where $F_l(\boldsymbol{x}_l) = \boldsymbol{w}_l * \boldsymbol{x}_l + \boldsymbol{b}_l$ denotes the output of the l^{th} layer before the gate function $g_l \in \{0, 1\}$. $\boldsymbol{w}_l \in \mathbb{R}^{C_{l+1} \times C_l \times K^l \times K^l}$ denotes the weights of the convolution kernels, and $*$ denotes the convolution operation. We omit the nonlinearity in Eq. 1 for clarity. Note that the gated function makes binary decisions (0 or 1), instead of predicting a continuous value to weightedly combine the output and the input of the layer l. This means, when $g_l = 0$, the convolution computation $F_l(\boldsymbol{x}_l)$ in layer l can be skipped to reduce computation cost.

Width-Dynamic Neural Architecture, such as [2, 7, 15, 32], employs a gate function to predict the activation status of each channel: $g_l \in \{0, 1\}^{C_{l+1}}$. The predicted activation status is then used to mask out the convolution operations on selective channels: $\boldsymbol{x}_{l+1} = F_l(\boldsymbol{x}_l, g_l) = (g_l \circ \boldsymbol{w}_l) * \boldsymbol{x}_l + g_l \circ \boldsymbol{b}_l$, where $\circ$ denotes the Hadamard product over the dimension of output channel C_{l+1}. When $g_{lc} = 0$, the computation of the c^{th} channel at the l^{th} layer could be skipped and the corresponding computation cost could thus be saved.

Gate Generation. In many works [18,28–30,40,44,45,49], the gate is often generated by a gating network $G_l(\cdot)$ as: $g_l = \begin{cases} 1, G_l(\boldsymbol{x}_l) \geq \tau \\ 0, G_l(\boldsymbol{x}_l) < \tau \end{cases}$, where $G_l(\boldsymbol{x}_l) \in (G_{\min}, G_{\max})$ is the estimated gating value of the l^{th} layer, and $\tau \in (G_{\min}, G_{\max})$ denotes the threshold. Note that, $(G_{\min}, G_{\max})$ indicates that $G_l(\cdot)$ is bounded, e.g., $G_{\min} = 0, G_{\max} = 1$ if sigmoid is adopted as the activation function of the gating network. For dynamic width networks, the gating network predicts a vector-based mask corresponding to the activation status of each channel, which could be constructed in a similar way: $g_{lc} = \begin{cases} 1, G_{lc}(\boldsymbol{x}_l) \geq \tau \\ 0, G_{lc}(\boldsymbol{x}_l) < \tau \end{cases}$. Therefore, to perform energy-oriented attack, we could perturb the input samples to raise the intermediate gating values to go beyond the threshold τ, which leads to the corresponding gates being activated and incurs extra computational cost to the dynamic neural networks.

3.2 Overall Objective of Computational Complexity Attack

In traditional accuracy-oriented attacks, to construct an adversarial example, a human imperceptible perturbation is created to modify a given input. A specific objective function (e.g., changing the predicted logits of an image classification network yet keeping the minimum amount of changes) is often constructed to guide the search for the perturbation. After iterative updates on the perturbations, the modified input could alter the predictions of the threatened models. Similarly, we can also construct perturbed input samples to invalidate the acceleration strategy of the dynamic neural architecture, such as raising the gating value to activate more gates and accordingly computes more layers or more channels. We detail this kind of attack below:

First, we initialize a specific perturbation $\boldsymbol{\delta} \in \mathbb{R}^{3 \times H \times W}$ and use it to modify the input image by:

$$\boldsymbol{x}_0' = \boldsymbol{x}_0 + \boldsymbol{\delta} \tag{2}$$

where H, W denote the height and width of the input image, $\boldsymbol{x}_0, \boldsymbol{x}_0' \in [0,1]^{3\times H\times W}$ denote the original input and modified input, respectively. The value of the input image should be in $[0,1]$. Hence, we follow Carlini *et al.* [6] to use $\tanh(\cdot) \in [-1,1]$ to refine Eq. 2 to force the modified input $\boldsymbol{x}_0' \in [0,1]^{3\times H\times W}$ by:

$$\boldsymbol{x}_0' = \frac{1}{2}\cdot(\tanh(\boldsymbol{x}_0+\boldsymbol{\delta})+1). \tag{3}$$

To make the perturbation $\boldsymbol{\delta}$ being able to increase the complexity of the network, we refer to Szegedy *et al.* [41] and formally define the objective as:

$$\min_{\boldsymbol{\delta}} \frac{1}{3HW}\|\frac{1}{2}\cdot(\tanh(\boldsymbol{x}_0+\boldsymbol{\delta})+1)-\boldsymbol{x}_0\|_2, \quad s.t. \;\; G_l(\boldsymbol{x}_0,\boldsymbol{\delta}) \geq \tau, \tag{4}$$

where $\|\cdot\|_2$ denotes the $L2$-norm, and $\frac{1}{3HW}\|\cdot\|_2$ denotes the mean-square error, which is used to minimize the deviation between the original input and the modified input to prevent them from being differentiated. The constraint is to guarantee the gating value G_l (or G_{lc} for width-dynamic neural architecture) being above the threshold to activate the execution of the l^{th} convolutional layer for more computational complexity. Here, we use $G_l(\boldsymbol{x}_0,\boldsymbol{\delta})$ as the gating function to indicate that it depends on both the input and the perturbation.

3.3 Complexity Loss

To increase the network complexity, we drive the gating value G_l to be larger than the threshold τ to satisfy the constraint: $G_l(\boldsymbol{x}_0,\boldsymbol{\delta}) \geq \tau$ in Eq. 4 to get the corresponding layer executed. The constraint can be considered as equivalent to $\max(\tau - G_l(\boldsymbol{x}_0,\boldsymbol{\delta}),0)=0$. We adopt Lagrangian relaxation to approximate Eq. 4 as:

$$\min_{\boldsymbol{\delta}} \frac{1}{3HW}\|\frac{1}{2}\cdot(\tanh(\boldsymbol{x}_0+\boldsymbol{\delta})+1)-\boldsymbol{x}_0\|_2 + \sum_l \lambda_l \cdot \max(\tau - G_l(\boldsymbol{x}_0,\boldsymbol{\delta}),0), \tag{5}$$

where λ_l is the chosen positive weight for the l^{th} layer. We define the right part in Eq. 5 as the Complexity Loss:

$$\mathcal{L}_{\text{C}}(\boldsymbol{x}_0,\boldsymbol{\delta}) = \sum_l \lambda_l \cdot \max(\tau - G_l(\boldsymbol{x}_0,\boldsymbol{\delta}),0), \tag{6}$$

which can rise all the deactivated gating values, i.e., $G_l(\boldsymbol{x}_0,\boldsymbol{\delta}) < \tau$, until being activated, i.e., being above the threshold τ. Then, we define λ_l as: $\lambda_l = \frac{C_l}{\sum_l C_l}$, where C_l denotes the computational complexity of the l^{th} layer, i.e., λ_l denotes the complexity proportion of l^{th} layer to all the convolutional layers. Using different λ_l for different gating values can reweight the losses of different convolutional layers according to their corresponding complexities. For example, when $C_0 = C_1 + C_2$, then the penalty for skipping the 0^{th} layer will be the same as the penalty for skipping both the 1^{th} layer and the 2^{th} layer.

For simplicity, we rewrite Eq. 5 to:

$$\min \|\frac{1}{2} \cdot \big[\tanh(\boldsymbol{x}_0 + \boldsymbol{\delta}) + 1\big] - \boldsymbol{x}_0\|_2 + 3HW \cdot \mathcal{L}_{\mathrm{C}}(\boldsymbol{x}_0, \boldsymbol{\delta}), \tag{7}$$

as the final objective. The Complexity Loss measures the computational complexity difference between the current state and the desirable state whose complexity is maximized. The final objective leads the modified input to increase the network complexity while keeping the deviation to the original input small.

For the width-dynamic neural architecture, the Complexity Loss becomes:

$$\mathcal{L}_{\mathrm{C}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = \sum_{l,c} \lambda_{lc} \cdot max(\tau - G_{lc}(\boldsymbol{x}_0, \boldsymbol{\delta}), 0), \tag{8}$$

where λ_{lc} is computed as the complexity proportion of the corresponding channel to the entire network.

4 GradAuto

In Sec. 3.3, we formulate the objective function as Eq. 7, where we minimize the combination of the overall magnitude of the perturbation $\boldsymbol{\delta}$ and the amount of gate value $G_l(\boldsymbol{x}_0, \boldsymbol{\delta})$ below the activation threshold τ. However, in practice, we found that directly optimizing the perturbation based on the gradient of Eq. 7 is suboptimal due to two main reasons: 1) The gradient becomes unstable and could change drastically when the status of the gate changes, as shown in plots A, B, and C of Fig. 1. 2) The gradient for bumping up the gate value of inactivated gates may disagree with the gradients to keep the activated gate activated, leading to previously activated gates deactivated, as shown in gate 1 in plot D of Fig. 1. These two issues impede the effective and efficient search of the optimal perturbation, leading to suboptimal attack performance as shown in our ablation studies in Sec. 5.3. To address these two issues, we propose two rectified based approaches to regularize the gradients, enabling faster and more effective energy-oriented attacks.

4.1 Rectified Magnitude of Gradient

Fig. 1 shows the gating value and the gradient during adversarial training. It can be observed in the sub-figures A and B that the magnitude of the gradient, i.e., $\|\boldsymbol{g}\|$, grows very fast at around 0^{th}, 100^{th} and 250^{th} iterations, which is the moment that the gating values of gate 1, gate 4, and gate 3 exceed the threshold (see sub-figure C) and we add the corresponding layer into our graph. As the result, the gradient maintains a low magnitude most of the time while occasionally grows abruptly, which leads to an unstable optimization process and sometimes even prevents the convergence of the Complexity Loss.

We suggest, there are two reasons for the gradient of dynamic neural architecture to be unstable during training: 1) The sudden change of the activation status

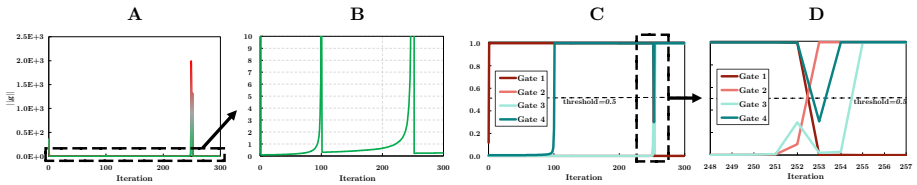

Fig. 1. Visualization of gradient and gating value. A and B show the magnitude of gradient of every training iteration. C and D show the change of gating value during adversarial training.

brings discontinuities to the dynamic neural networks, leading to additional layers involved into the computation graph and drastic jumps of the gradients. 2) The range of the derivative of gating network is very large which will significantly scale up or down the gradient propagated backward. A bounded and monotonic activation function, such as sigmoid or hyperbolic tangent, is often adopted in gating network to restrict the range of its output value while their derivatives increase (or decreases) extremely fast. More specifically, the derivative of sigmoid is:

$$\sigma'(x) = \frac{1}{1+e^{-x}} \cdot (1 - \frac{1}{1+e^{-x}}) = \sigma(x) \cdot (1 - \sigma(x)), \tag{9}$$

where $\sigma'(0) = 2.5 \times 10^{-1}$, while $\sigma'(-5) = 6.6 \times 10^{-3}$.

To stabilize the gradient, we resort to limit the change of the Complexity Loss to meet Lipschitz continuity as

$$d_{\mathcal{L}_\mathrm{C}}(\mathcal{L}_\mathrm{C}(\boldsymbol{\delta}_1), \mathcal{L}_\mathrm{C}(\boldsymbol{\delta}_2)) \leq K d_{\boldsymbol{\delta}}(\boldsymbol{\delta}_1, \boldsymbol{\delta}_2), \tag{10}$$

where $K \in \mathbb{R}_{>0}$ denotes the Lipschitz constant, and $d(\cdot)$ denotes the metric on the corresponding space of $\boldsymbol{\delta}$ or $\mathcal{L}_\mathrm{C}$. Hence, Eq. 10 can be written as:

$$\frac{|\mathcal{L}_\mathrm{C}(\boldsymbol{\delta}_1) - \mathcal{L}_\mathrm{C}(\boldsymbol{\delta}_2)|}{\|\boldsymbol{\delta}_1 - \boldsymbol{\delta}_2\|_2} \leq K. \tag{11}$$

According to the mean value theorem, Eq. 11 is valid if and only if the limit

$$\lim_{\|\Delta\boldsymbol{\delta}\|_2 \to 0} \frac{|\mathcal{L}_\mathrm{C}(\boldsymbol{\delta} + \Delta\boldsymbol{\delta}) - \mathcal{L}_\mathrm{C}(\boldsymbol{\delta})|}{\|\Delta\boldsymbol{\delta}\|_2} \leq K. \tag{12}$$

holds, where $\Delta\boldsymbol{\delta}$ represents any change of the perturbation $\boldsymbol{\delta}$. Eq. 12 can be formulated by directional derivative as:

$$\lim_{\|\Delta\boldsymbol{\delta}\|_2 \to 0} \frac{|\langle \nabla\mathcal{L}_\mathrm{C}(\boldsymbol{\delta}), \Delta\boldsymbol{\delta} \rangle|}{\|\Delta\boldsymbol{\delta}\|_2} \leq K, \tag{13}$$

$$\lim_{\|\Delta\boldsymbol{\delta}\|_2 \to 0} \frac{\|\nabla\mathcal{L}_\mathrm{C}(\boldsymbol{\delta})\|_2 \cdot \|\Delta\boldsymbol{\delta}\|_2 \cdot |\cos\theta|}{\|\Delta\boldsymbol{\delta}\|_2} \leq K, \tag{14}$$

$$\|\nabla\mathcal{L}_\mathrm{C}(\boldsymbol{\delta})\|_2 \cdot |\cos\theta| \leq K, \tag{15}$$

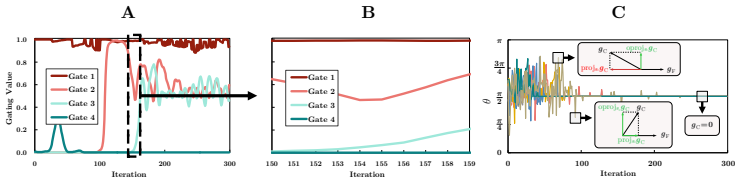

Fig. 2. Visualization of gradient and gating value after being rectified. A and B show the change of gating value during training, where the rectified magnitude of gradient is adopted (A and B can be compared to C and D in Fig. 1). C shows the change of the included angle θ between Finished Gradient and Complexity Gradient, and the gradient projection where the curves in different color represent different input image samples.

where $\langle \cdot, \cdot \rangle$ denotes inner product, and θ denotes the angle between the gradient $\nabla \mathcal{L}_{\mathrm{C}}(\boldsymbol{\delta})$ and the change $\Delta \boldsymbol{\delta}$ of the perturbation, which equals to zero when the $\cos\theta$ is maximized. Therefore, the sufficient condition of Eq. 15 is:

$$\left\| \frac{\nabla \mathcal{L}_{\mathrm{C}}(\boldsymbol{\delta})}{K} \right\|_2 \leq 1, \tag{16}$$

Hence, we rectify the Complexity Loss in Eq. 6 to satisfy the sufficient condition Eq. 16 as:

$$\mathcal{L}_{\mathrm{C}}(\boldsymbol{\delta}) = K \cdot \frac{\sum_l \lambda_l \cdot max(\tau - G_l(\boldsymbol{x}_0, \boldsymbol{\delta}), 0)}{\left\| \nabla \sum_l \lambda_l \cdot max(\tau - G_l(\boldsymbol{x}_0, \boldsymbol{\delta}), 0) \right\|_2}, \tag{17}$$

This rectified version of the Complexity Loss is K-Lipschitz continuous, where we can adjust K to control the slope of the Complexity Loss as well as the magnitude of the gradient.

We further visualize the gradients and the gating values of the rectified Complexity Loss in Fig 2. The sub-figures A and B plot the change of the gating value under the case of $K = 1$. Compared to the sub-figures C and D in Fig 1, our AutoGrad damps the growth of the gating value and leaves room for the gates to compete with each other during the loss convergence, i.e., the activated gates would not drop to the bottom in one shot when the value of inactivated gates are growing towards the threshold. Once the values for the activated gates drop below the threshold, they will be involved into our Complexity Loss (Eq. 17) and compete against other inactivated gates.

4.2 Rectified Direction of Gradient

The Complexity Loss only penalizes the inactivated gates (gate with value below the threshold) to drive them to be activated, while the already activated gates (gates with values above the threshold) are excluded. However, the gradients to boost the inactivated gates' values may not always agree with the gradients to make the activated gates keep activated. Hence some of the activated gates' value

may drop and become inactivated again. This kind of disagreement will drive the gates switching back and forth between being activated and inactivated, as shown in sub-figures D of Fig. 1. While the Gate 2 increases to above the threshold, the Gate 1 drops to below the threshold and becomes inactivated. In this case, they almost cannot be activated in the same time, which limits the total number of activated gates. Below we study two ways to address this problem.

Involving Activated Gates in the Complexity Loss. In this case, we modify the Complexity Loss to encourage the gating value to rise even after activation:

$$\mathcal{L}_{\mathrm{F}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = \sum_l \lambda_l \cdot (\tau - G_l(\boldsymbol{x}_0, \boldsymbol{\delta})), \tag{18}$$

However, we empirically find this modification effectively leads to a tighter constraint:

$$G_l(\boldsymbol{x}_0, \boldsymbol{\delta}) = G_{\max} \tag{19}$$

for Eq. 4, while the $L2$-norm also constrains the amount of the perturbation $\boldsymbol{\delta}$. Combining these two constraints, the solution space is extremely limited. As the result, keeping the activated gate rising induces degeneracy of the solution space further, and poses greater challenges to the perturbation search process.

Gradient Projection for Complexity Loss. Keeping rising the activated gate can prevent them from switching back to some extent but always keeping them rising impedes the convergence of Complexity Loss. Moreover, encouraging higher activation values for those activate gates does not further contribute to the complexity increase, because their corresponding layer has already been counted. Hence, we further propose Finished Gradient Masking to keep them **only from dropping**. Firstly, we additionally compute the Finished Loss for those activated gates as:

$$\mathcal{L}_{\mathrm{F}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = -\sum_l \lambda_l \cdot max(G_l(\boldsymbol{x}_0, \boldsymbol{\delta}) - \tau, 0) \tag{20}$$

The Finished Loss $\mathcal{L}_{\mathrm{F}}(\boldsymbol{x}_0, \boldsymbol{\delta})$ provides us the Finished Gradient $\boldsymbol{g}_{\mathrm{F}} \in \mathbb{R}^{3\times H\times W}$ for the perturbation $\boldsymbol{\delta}$, which is the direction to further increase the gate value for activated gates:

$$\boldsymbol{g}_{\mathrm{F}} = \nabla_{\boldsymbol{\delta}} \mathcal{L}_{\mathrm{F}}(\boldsymbol{x}_0, \boldsymbol{\delta}). \tag{21}$$

Secondly, we use the Complexity Loss to calculate the Complexity Gradient $\boldsymbol{g}_{\mathrm{C}} \in \mathbb{R}^{3\times H\times W}$ by

$$\boldsymbol{g}_{\mathrm{C}} = \nabla_{\boldsymbol{\delta}} \mathcal{L}_{\mathrm{C}}(\boldsymbol{x}_0, \boldsymbol{\delta}). \tag{22}$$

Thirdly, we project the Complexity Gradient onto the Finished Gradient by

$$\mathrm{proj}_{\boldsymbol{g}_{\mathrm{F}}} \boldsymbol{g}_{\mathrm{C}} = \frac{\langle \boldsymbol{g}_{\mathrm{C}}, \boldsymbol{g}_{\mathrm{F}} \rangle}{\|\boldsymbol{g}_{\mathrm{F}}\|_2} \frac{\boldsymbol{g}_{\mathrm{F}}}{\|\boldsymbol{g}_{\mathrm{F}}\|_2}, \tag{23}$$

and accordingly calculate the rejection of the Complexity Gradient from the Finished Gradient as:

$$\text{oproj}_{\boldsymbol{g}_\text{F}} \boldsymbol{g}_\text{C} = \boldsymbol{g}_\text{C} - \text{proj}_{\boldsymbol{g}_\text{F}} \boldsymbol{g}_\text{C}, \tag{24}$$

whose direction is orthogonal to the Finished Gradient. Hence updating with it dose not affect the gates that have already been activated. Finally, we rectify the direction of Complexity Gradient as:

$$\boldsymbol{g}'_\text{C} = \begin{cases} \text{oproj}_{\boldsymbol{g}_\text{F}} \boldsymbol{g}_\text{C} + \text{proj}_{\boldsymbol{g}_\text{F}} \boldsymbol{g}_\text{C}, & \left\langle \text{proj}_{\boldsymbol{g}_\text{F}} \boldsymbol{g}_\text{C}, \boldsymbol{g}_\text{F} \right\rangle \geq 0, \\ \text{oproj}_{\boldsymbol{g}_\text{F}} \boldsymbol{g}_\text{C}, & \left\langle \text{proj}_{\boldsymbol{g}_\text{F}} \boldsymbol{g}_\text{C}, \boldsymbol{g}_\text{F} \right\rangle < 0 \end{cases}, \tag{25}$$

which indicates that when the projection is opposite to the direction of Finished Gradient, it will be removed and only the orthogonal component of the Complexity Gradient will be updated. With this design, we update the perturbation to activate more gates and effectively avoids the activated gate from being deactivated again.

The sub-figure C in Fig. 2 visualizes the change of the angle θ between the Finished Gradient and the Complexity Gradient during training. When the angle is obtuse, the Complexity Gradient conflicts with the Finished Gradient, i.e., updating alongside the vanilla Complexity Gradient drags the activated gates toward the threshold. As shown in the plot, such conflicts occur frequently during training which drives the gate switching back and forth. Our approach of avoiding the conflict with gradient projection can be considered as a greedy algorithm to push the gate values of the remaining inactivated gates to as high as possible. Therefore, we adopt it at the later stage of the training to activate the rest of inactivated gates as much as possible.

5 Experiments

We validate our approach by using it to attack popular dynamic depth network (SkipNet [45] and SACT [14]) and dynamic width network (ManiDP [42]) on CIFAR-10 and ImageNet. For the experiments on dynamic depth network, we attack SkipNet with two different settings: SkipNet+SP and SkipNet+SP+HRL, where +SP and +HRL indicate whether supervised pre-training or hybrid reinforcement learning were used following [45]. SkipNet+SP+HRL achieves better efficiency compared to SkipNet+SP. Since ILFO only attacks SkipNet+SP, we further re-implement their code on the SkipNet+HRL and compare it with our GradAuto.

Metric We follow the previous work [20] and evaluate the effectiveness of our attacks. We report the average percentage of the recovery in the dynamic neural network reduced FLOPs during inference following [20]. Dynamic neural networks reduced the total computation complexities at inference stage and we measure the amount of the saving computations that are invalidated by our attacks.

To measure the Quality, we adopt **peak signal-to-noise ratio (PSNR)** as the metric, which approximates the human perceptual quality and is commonly employed to evaluate image quality [24]. The PSNR can be defined as: $\text{PSNR} = 10 \cdot \log_{10}(\frac{\text{MAX}_\text{I}^2}{\text{MSE}})$, where MAX_I is the maximum possible pixel value of the image. For an original m×n image I and its corresponding adversarial example K, **mean squared error(MSE)** can be calculated by: $\text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$.

5.1 Attack on Dynamic Depth Network

We compare our approach with the state of the art method ILFO [20]. To ensure a fair comparison, we follow the experimental settings in [20]. Images from ImageNet and CIFAR-10 are converted into 224×224×3 and 32×32×3, respectively.

Quantitative Comparison We present the comparison results in Table 1. On both CIFAR-10 and ImageNet, our GradAuto invalidates **all the reduced FLOPs**, outperforming the ILFO baseline by 15.71% & 18.64% for the SkipNet and 27.51% & 8.94% for SACT. In these experiments, our GradAuto attacks demolish the adaptive mechanism of SkipNet and SACT completely, effectively reduced them to their static counterparts, which shows the effectiveness of the proposed GradAuto.

To further evaluate the limits of our GradAuto, we additionally design and compare the ILFO on the more powerful SkipNet+HRL, i.e., the SkipNet with hybrid reinforcement learning. The original complexity reduction of SkipNet and SkipNet+HRL are 15.08% and 29.94% respectively. As shown in Table 1, our GradAuto outperforms ILFO by a large margin on attacking the SkipNet-HRL, effectively recovers 70% more reduced FLOPs compared to ILFO.

Qualitative Comparison Fig. 3 visualizes the modified input images generated by ILFO and ours GradAuto based on SkipNet-HRL. It can be observed that our method improves attack performance with lower deviation compared to ILFO. ILFO greatly changes the contrast of the image, while our GradAuto is more authentic to the original image. Results with different Lipschitz constant K show that on the our method generates less perceptible noise at the background.

Table 1. Average recovery of the reduced FLOPs by three dynamic neural architectures: SkipNet, SACT, and ManiDP on the two datasets: CIFAR-10 and ImageNet, respectively.

Dataset	CIFAR-10			ImageNet			
Model	SkipNet (%)	SACT (%)	ManiDP (%)	SkipNet (%)	SACT (%)	SkipNet+HRL (%)	ManiDP (%)
ILFO [20]	84.29	72.49	80.3	81.36	91.06	46.9	85.6
GradAuto(ours)	100	100	100	100	100	88.9	100

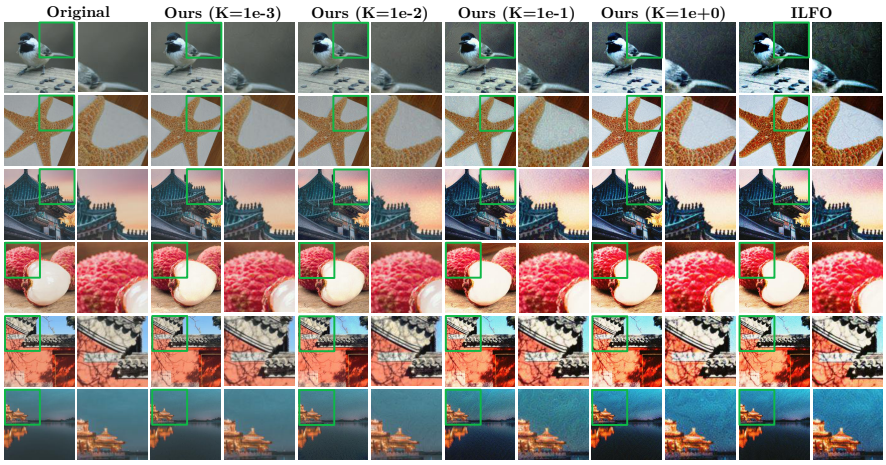

Fig. 3. The modified input images generated by our AutoGrad under different Lipschitz constant K of Lipschitz continuity. The model for the experiment is SkipNet-HRL.

5.2 Attack on Dynamic Width Network

We next evaluate our attack on the dynamic width network ManiDP [42]. The experimental setting of attack on ManiDP is same with the attack on SkipNet. The original computation reduction of ManiDP on ImageNet is 49%. We evaluate our approach on the ImageNet and CIFAR-10. As shown in Table 1, our GradAuto outperforms ILFO and recover ManiDP reduced FLOPs by 100% for all samples.

5.3 Ablation Study

To further validate the effect of a few important design choices of our method, we conduct ablation studies on the gains of rectified gradient, the selection of the Lipschitz constant as well as the design of rectified direction of gradient.

Table 2. Comparison for different methods. Baseline denotes ILFO. RD and RM denote Rectified Direction and Rectified Magnitude, respectively. GradAuto denotes the combination of Rectified Direction and Rectified Magnitude.

Method	Baseline	RD (Ours)	RM (Ours)	GradAuto (Ours)
Complexity Increase	46.9	52.3	86.9	88.9

Rectified Gradient. Table 2 evaluates the gains brought by the Rectified Direction and the Rectified Magnitude. Both Rectified Direction and Rectified

Table 3. Result comparison among different Lipschitz constant K of Lipschitz continuity. Recovery denote the recovery of reduced FLOPs by the dynamic neural architecture. Baseline denotes ILFO [20]. PSNR measures the deviation between the original input and the modified input. The model for the experiment is SkipNet-HRL.

K	Baseline	$1e-3$	$1e-2$	$1e-1$	$1e+0$	$1e+1$	$1e+2$
Recovery (%)	46.9	71.0	85.2	87.0	88.9	88.9	88.9
PSNR	54.39	78.13	67.60	57.57	54.54	53.92	53.87

Magnitude provide better attacking performance, incurring 5.4% and 40% more computation complexity.

The Setting of K-Lipschitz. Table 3 lists the performance of Complexity Attack and the deviation of modified input images under different Lipschitz constant. A lower Lipschitz constant reduces the PSNR between the original input images and the modified input images but also drops the performance of Complexity Attack (measured by complexity increase). Note that, when the Lipschitz constant $K \in [1e-3, 1e-0]$, our AutoGrad significantly improves the performance of Complexity Attack while achieves extremely high PSNR. It can be observed that when setting the Lipschitz constant as $K = 1e-3$, the modified inputs generated by our method are hardly distinguishable from the original input images while improves the attack performance by over 20.8% compared to ILFO.

A higher Lipschitz constant accelerates the convergence of Complexity Loss which activates more gates and bring better attack performance. Meanwhile, the perturbations also becomes more noticeable. Finally, we can adjust the Lipschitz constant based on the trade off between the perceptiveness and the performance of the Complexity Attack in response to distinct needs.

6 Conclusion

In this paper, we investigate the robustness of dynamic neural network in terms of computational efficiency. We construct a framework for attacking the dynamic depth/width networks and also identify two key issues in the gradient space causing the limit success of the prior attack. First, the magnitude of the gradient is not stable. Sudden spikes in gradients make the network difficult to train. Second, gradient of different gates may have conflicting directions. To address these two issues, we proposed a new attack approach GradAuto and empirically demonstrate that it could invalidate 100% of reduced FLOPs of both dynamic depth and dynamic width model while keeping the perturbation imperceptible.

Acknowledgments This work is supported by National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-100E-2020-065), MOE Tier 1 Grant, and SUTD Startup Research Grant. The research is also supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence **39**(12), 2481–2495 (2017)
2. Bejnordi, B.E., Blankevoort, T., Welling, M.: Batch-shaping for learning conditional channel gated networks. arXiv preprint arXiv:1907.06627 (2019)
3. Bengio, E., Bacon, P.L., Pineau, J., Precup, D.: Conditional computation in neural networks for faster models. arXiv preprint arXiv:1511.06297 (2015)
4. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
5. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
6. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. IEEE (2017)
7. Chen, J., Zhu, Z., Li, C., Zhao, Y.: Self-adaptive network pruning. In: International Conference on Neural Information Processing. pp. 175–186. Springer (2019)
8. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
9. Cheng, J., Wang, P., Li, G., Hu, Q., Lu, H.: Recent advances in efficient computation of deep convolutional neural networks. arXiv preprint arXiv:1802.00939 (2018)
10. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282 (2017)
11. Cho, K., Bengio, Y.: Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. arXiv preprint arXiv:1406.7362 (2014)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
13. Eigen, D., Ranzato, M., Sutskever, I.: Learning factored representations in a deep mixture of experts. arXiv preprint arXiv:1312.4314 (2013)
14. Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R.: Spatially adaptive computation time for residual networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1039–1048 (2017)
15. Gao, X., Zhao, Y., Dudziak, Ł., Mullins, R., Xu, C.z.: Dynamic channel pruning: Feature boosting and suppression. arXiv preprint arXiv:1810.05331 (2018)
16. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
17. Graves, A.: Adaptive computation time for recurrent neural networks. arXiv preprint arXiv:1603.08983 (2016)
18. Guo, Q., Yu, Z., Wu, Y., Liang, D., Qin, H., Yan, J.: Dynamic recursive neural network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5147–5156 (2019)

19. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
20. Haque, M., Chauhan, A., Liu, C., Yang, W.: Ilfo: Adversarial attack on adaptive neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
21. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
23. Hong, S., Kaya, Y., Modoranu, I.V., Dumitras, T.: A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=9xC2tWEwBD
24. Hore, A., Ziou, D.: Image quality metrics: Psnr vs. ssim. In: 2010 20th international conference on pattern recognition. pp. 2366–2369. IEEE (2010)
25. Hua, W., Zhou, Y., De Sa, C.M., Zhang, Z., Suh, G.E.: Channel gating neural networks. Advances in Neural Information Processing Systems **32** (2019)
26. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
27. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural computation **3**(1), 79–87 (1991)
28. Jin, Q., Yang, L., Liao, Z.: Adabits: Neural network quantization with adaptive bit-widths. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2146–2156 (2020)
29. Leroux, S., Molchanov, P., Simoens, P., Dhoedt, B., Breuel, T., Kautz, J.: Iamnn: Iterative and adaptive mobile neural network for efficient image classification. arXiv preprint arXiv:1804.10123 (2018)
30. Li, C., Wang, G., Wang, B., Liang, X., Li, Z., Chang, X.: Dynamic slimmable network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8607–8617 (2021)
31. Li, Y., Song, L., Chen, Y., Li, Z., Zhang, X., Wang, X., Sun, J.: Learning dynamic routing for semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8553–8562 (2020)
32. Lin, J., Rao, Y., Lu, J., Zhou, J.: Runtime neural pruning. Advances in neural information processing systems **30** (2017)
33. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
34. Liu, C., Wang, Y., Han, K., Xu, C., Xu, C.: Learning instance-wise sparsity for accelerating deep models. arXiv preprint arXiv:1907.11840 (2019)
35. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
36. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10012–10022 (2021)

37. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. arXiv preprint arXiv:2201.03545 (2022)
38. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
39. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
40. Shen, J., Wang, Y., Xu, P., Fu, Y., Wang, Z., Lin, Y.: Fractional skipping: Towards finer-grained dynamic cnn inference. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5700–5708 (2020)
41. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
42. Tang, Y., Wang, Y., Xu, Y., Deng, Y., Xu, C., Tao, D., Xu, C.: Manifold regularized dynamic network pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5018–5028 (2021)
43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
44. Veit, A., Belongie, S.: Convolutional networks with adaptive inference graphs. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–18 (2018)
45. Wang, X., Yu, F., Dou, Z.Y., Darrell, T., Gonzalez, J.E.: Skipnet: Learning dynamic routing in convolutional networks. In: Proceedings of the European Conference on Computer Vision (ECCV) (September 2018)
46. Wang, Y., Shen, J., Hu, T.K., Xu, P., Nguyen, T., Baraniuk, R., Wang, Z., Lin, Y.: Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference. IEEE Journal of Selected Topics in Signal Processing **14**(4), 623–633 (2020)
47. Xia, W., Yin, H., Dai, X., Jha, N.K.: Fully dynamic inference with deep neural networks. IEEE Transactions on Emerging Topics in Computing (2021)
48. Yang, B., Bender, G., Le, Q.V., Ngiam, J.: Condconv: Conditionally parameterized convolutions for efficient inference. Advances in Neural Information Processing Systems **32** (2019)
49. Yu, H., Li, H., Shi, H., Huang, T.S., Hua, G., et al.: Any-precision deep neural networks. arXiv preprint arXiv:1911.07346 **1** (2019)