GradMDM: Adversarial Attack on Dynamic Networks

Jianhong Pan, Lin Geng Foo, Qichen Zheng, Zhipeng Fan, Hossein Rahmani, Qiuhong Ke, Jun Liu

Abstract—Dynamic neural networks can greatly reduce computation redundancy without compromising accuracy by adapting their structures based on the input. In this paper, we explore the robustness of dynamic neural networks against *energy-oriented attacks* targeted at reducing their efficiency. Specifically, we attack dynamic models with our novel algorithm GradMDM. GradMDM is a technique that adjusts the direction and the magnitude of the gradients to effectively find a small perturbation for each input, that will activate more computational units of dynamic models during inference. We evaluate GradMDM on multiple datasets and dynamic models, where it outperforms previous energy-oriented attack techniques, significantly increasing computation complexity while reducing the perceptibility of the perturbations.

1 Introduction

In recent years, research on Deep Neural Networks (DNNs) has seen tremendous progress, with DNNs being widely developed for various areas, such as image classification [1], [2], [3], [4], [5], segmentation [6], [7], [8], [9], [10] and object detection [10], [11], [12], [13], [14]. However, as DNNs continue to attain better and better performance, they have also been steadily increasing in size, even reaching up to the scale of billions of parameters [15]. These computation-heavy models are highly challenging to be deployed on embedded and mobile devices, motivating research on the development of efficient models [16], [17] to accelerate the inference process.

To this end, dynamic neural networks [18] offer a promising strategy for accelerating the inference process, by skipping the redundant computations on-the-fly. This is typically achieved by introducing gating mechanisms [18] to adaptively execute only a subset of computational units, conditioned on the input sample. Thus, dynamic neural networks provide a better trade-off between accuracy and efficiency than their static counterparts [18], as shown throughout many works [19], [20], [21].

Despite the appeal and rising popularity of dynamic neural networks, the robustness of their mechanisms and efficiency gains is still an important issue that requires more studies. As shown in ILFO [22], dynamic neural networks are highly susceptible to *energy-oriented attacks*, which are adversarial attacks aiming at boosting the energy consumption and computational load of the dynamic neural network. These energy-oriented attacks can drastically reduce the number of FLOPs (floating-point operations) saved by the dynamic mechanisms. Importantly, such research on energy-oriented attacks broadens our understanding of dynamic mechanisms, and contributes an alternative point-of-view studying the potential risks of dynamic networks.

- *J. Pan, L. G. Foo, Q. Zheng and J. Liu are with SUTD. Z. Fan is with NYU. H. Rahmani is with Lancaster University. Q. Ke is with Monash University. (J. Pan and L. G. Foo contributed equally.)*

In this paper, we provide insights on energy-oriented attacks and identify two key pitfalls of the existing method [22], as shown in Sec. 4.2 and Sec. 4.3. Firstly, ILFO [22] attacks gates to lower a loss function (a Complexity Loss), but the direct minimization of this loss might not be enough to fully maximize the number of activated gates for a given "allowable" amount of perturbation to the image. Secondly, we observe difficulties in the joint optimization of the many gates' losses, that makes it difficult to simultaneously keep many gates activated. We attribute this to the presence of *conflicting gradients* between the gates, which makes it hard to activate new gates without interfering with already-activated gates.

To mitigate the above-mentioned issues, we propose GradMDM, a **Grad**ient **M**agnitude and **D**irection **M**anipulation method for energy-oriented attacks against dynamic networks. (1) We introduce a Power Loss to modify the magnitudes of the losses across various gates. This loss prioritizes the optimization of gates in a principled manner – to find an optimal spot with more activated gates along the Pareto Frontier – which allows for activation of more gates while keeping the input perturbation imperceptible. (2) Next, we propose Complexity Gradient Masking (CGM), a technique that adjusts the directions of the Complexity Gradient to reduce gradient conflicts among gates. It involves a gradient projection process to rectify the gradient directions between activated and deactivated gates such that there are less conflicts between them, thus making it easier to activate new gates without negatively affecting already-activated gates. (3) We demonstrate the efficacy of GradMDM on multiple dynamic neural networks across various datasets, where GradMDM effectively increases computations with less perceptible perturbations across all settings.

2 Related Work

Dynamic Neural Networks [19], [20], [21], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33] achieve efficiency while maintaining good accuracy by dynamically adjusting model architectures to allocate appropriate computation conditioned on each input sample. This reduces redundant computations on those "easy" samples, improving the inference efficiency [18]. *Dynamic depth networks* [19], [20], [27], [33] adjust their architectures to use only activated layers to conduct inference, and often achieve efficient inference in one of two ways: early termination or conditional skipping. *Early termination networks* [20], [27], [33] are able to terminate their inference process early. For instance, Adaptive Computation Time (ACT) [27] augments an RNN with a "halting score" that adaptively terminates the recurrent computations to reduce cost. SACT [20] extends this concept to ResNets for vision tasks by applying ACT to each residual block group, allowing for an early exit. Another early termination network is the Shallow-deep network [33] which uses internal classifiers to facilitate confidence-based early exits. On the other hand, *conditional skipping models* [19], [23], [29] can decide to skip layers or blocks based on each input. SkipNet [19] assigns a gating module to each convolutional block in CNNs, which decides whether to execute or skip it, and is trained via reinforcement learning. *Dynamic width networks* [21], [32], [34] selectively activate multiple components within the same layer, such as channels and neurons, based on each instance. Early studies [35], [36], [37] achieve dynamic width by adaptively controlling the activation of neurons or parameters, e.g., via stochastic gating units [35], [36]. Subsequently, some works explore dynamic channel pruning in CNNs [21], [34], [38], which selectively activates convolutional channels. For instance, ManiDP [21] uses the relationship between samples (i.e., manifold information) to identify and dynamically remove redundant channels or filters.

In this paper, we propose a method to generate adversarial samples to perform energy-oriented attacks. We showcase the efficacy of our method on both the dynamic depth structures (SkipNet [19] and SACT [20]) as well as the dynamic width structures (ManiDP [21]).

Energy-oriented Attacks [22], [39] focus on delaying the inference speed of the target dynamic model, by incurring more computations within the model and increasing the energy consumption of inference. A successful energy-oriented attack will leave the dynamic neural network no longer efficient, therefore defeating the purpose of using them in the first place. Despite its significance, research into adversarial attacks on dynamic neural networks is quite limited. ILFO [22] is the first work investigating the robustness of dynamic neural networks. Deepsloth [39] attacks the early termination-based dynamic neural nets with an adversarial example crafting technique.

In this work, we show that directly searching for adversarial examples based on gradients, as done in previous works, could be sub-optimal. To address this, we propose two remedies to rectify the gradients, leading to improved adversarial performance with less perceptible perturbations.

3 Background

Gate Mechanism. In many works [19], [23], [41], [42], [43], [44], [45], [46] with dynamic depth or dynamic width architectures, a gating network $G_i(\cdot)$ with a sigmoid activation function is often used to generate gating values for gate g_i as follows:

$$g_i = \begin{cases} 1, & G_i(\boldsymbol{x}) \geq \tau, \\ 0, & G_i(\boldsymbol{x}) < \tau, \end{cases} \tag{1}$$

where $G_i(\boldsymbol{x}) \in [0, 1]$ is the estimated gating value of the i^{th} gate based on the input x, and $\tau \in (0, 1)$ denotes the threshold. If $g_i = 1$, the gate will execute the optional computations, and if $g_i = 0$, the gate will skip the optional computations. We remark that g_i can generally refer to the gating value of a gate in a dynamic depth network (e.g., for layer activation) or a dynamic width network (e.g., for channel activation), and so attacks on g_i can work for both dynamic depth and dynamic width architectures.

When performing an energy-oriented attack, our task is generally to make as many g_i produce values of 1 as possible. This is done by perturbing the input sample x to raise the gating values to go beyond the threshold τ, which leads to the corresponding gates being *activated*, incurring additional computational cost to the dynamic neural networks. To clarify, in this work, we take "activated" gates (i.e., $g_i = 1$) to mean that the gate is executing the optional computations and incurring higher computational costs.

4 Method: GradMDM Attack

In this section, we first formulate a Baseline Loss (Sec. 4.1) from the ground up, which can be used to conduct energy-oriented attacks against dynamic architectures. Then, we propose two techniques involved in our GradMDM attack that can improve upon the Baseline Loss. **1)** For the sake of achieving a better result, the attacks on some gates might be more important than others. Thus, we introduce a Power Loss (Sec. 4.2) that prioritizes the attacking of some gates over others to maximize the number of activated gates. **2)** As it might be difficult to jointly optimize many gates simultaneously (i.e., gradient conflicts might occur between different gates' losses), we propose our Complexity Gradient Masking (CGM) method (Sec. 4.3) to rectify the directions of gradients.

4.1 Ground Up Formulation of Baseline Loss for Computational Complexity Attack

Following previous works [22], [39], we construct an adversarial sample by creating a human-imperceptible perturbation to modify a given input. By conducting iterative updates using a carefully designed objective function, perturbations can be optimized to alter the predictions (i.e. gating values) of the threatened dynamic models and invalidate their acceleration strategy. We detail this kind of attack below.

Computational Complexity Attack. First, we initialize a specific perturbation $\boldsymbol{\delta} \in \mathbb{R}^{3\times H\times W}$ and use it to modify the input image as: $\boldsymbol{x}_0' = \boldsymbol{x}_0 + \boldsymbol{\delta}$, where H, W denote the height and width of the input image. $\boldsymbol{x}_0, \boldsymbol{x}_0' \in [0, 1]^{3\times H\times W}$ respectively denote the original input and modified input. We then follow [47] to use $\tanh(\cdot) \in [-1, 1]$, so that the pixel values of the modified input $\boldsymbol{x}_0'$ are within $[0, 1]$ similar to the original input $\boldsymbol{x}_0$, as follows:

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

To optimize the perturbation $\boldsymbol{\delta}$ for increasing the complexity of the network, we refer to Szegedy *et al.* [48] and formally define the objective and constraints as:

$$\text{minimize}_{\boldsymbol{\delta}} \left\| \frac{1}{2} \cdot (\tanh(\boldsymbol{x}_0 + \boldsymbol{\delta}) + 1) - \boldsymbol{x}_0 \right\|_2^2 \tag{3}$$

$$s.t.\ \ G_i(\boldsymbol{x}_0, \boldsymbol{\delta}) \geq \tau, \tag{4}$$

where $\|\cdot\|_2$ denotes the $L2$-norm, and its objective is to minimize the deviation between the original input and the modified input to prevent them from being easily distinguishable. The constraint enforces the gating value G_i to be above the threshold, to activate the execution of the i^{th} gate for more computational complexity. Here, we use $G_i(\boldsymbol{x}_0, \boldsymbol{\delta})$ as the gating function to indicate that it depends on both the input and the perturbation.

Baseline Loss. As it is difficult to optimize under the constrained formulation in Eq. 3, we can instead optimize our perturbations $\boldsymbol{\delta}$ using a loss $\mathcal{L}_{\text{relaxed}}$ derived from the relaxation of Eq. 3:

$$\mathcal{L}_{\text{relaxed}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = \gamma \left\| \frac{1}{2} \cdot (\tanh(\boldsymbol{x}_0 + \boldsymbol{\delta}) + 1) - \boldsymbol{x}_0 \right\|_2^2 - \sum_{i=1}^{N} (G_i(\boldsymbol{x}_0, \boldsymbol{\delta}) - \tau), \tag{5}$$

where γ is the weight of the first term, and N represents the total number of gates in the dynamic network. $\mathcal{L}_{\text{relaxed}}$ in Eq. 5 can be applied as a loss on the gate outputs, and backpropagated to optimize the perturbations $\boldsymbol{\delta}$. Moreover, the second term in Eq. 5 penalizes the outputs of *all* gates $\{G_i\}_{i=1}^N$. To further streamline the loss, we modify the second term such that losses only apply to gates $G_i(\boldsymbol{x}_0, \boldsymbol{\delta}) < \tau$. Thus, the Baseline Loss $\mathcal{L}_{\text{base}}$ is as follows:

$$\begin{aligned}\mathcal{L}_{\text{base}}(\boldsymbol{x}_0, \boldsymbol{\delta}) &= \gamma \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{C}} \\ &= \gamma \left\| \frac{1}{2} \cdot (\tanh(\boldsymbol{x}_0 + \boldsymbol{\delta}) + 1) - \boldsymbol{x}_0 \right\|_2^2 - \sum_{i=1}^{N} \min(G_i(\boldsymbol{x}_0, \boldsymbol{\delta}), \tau),\end{aligned} \tag{6}$$

which optimizes only the deactivated gates, so that the loss can ignore already-activated gates, and thus focuses more on minimizing the image perturbation. The first term $\mathcal{L}_{\text{MSE}}$ can be considered to be an Imperceptibility Loss which minimizes the visibility of perturbations, while the second term $\mathcal{L}_{\text{C}}$ is a Complexity Loss that maximizes the computational complexity of the dynamic network by activating the gates. We note that ILFO [22] uses the same mechanism as Eq. 6 to perform energy-oriented attacks.

4.2 Rectified Loss Magnitudes

An issue with the Baseline Loss is that directly optimizing using the Complexity Loss might not result in the maximum number of gates activated, because the Complexity Loss fails to consider the *overall optimal balance between the gating values* during optimization of this loss. Below, we first explain the problem in more detail by treating our task as a multi-objective optimization problem. Then, through the lens of the Pareto Frontier [49], we propose a Power Loss that can balance the optimization of gating values along the boundary such that more gates can be activated.

Pareto Frontier. In multi-objective optimization, a Pareto Frontier refers to a set of optimal solutions, where no single objective can be optimized further without making another objective worse off. In our scenario, as we are simultaneously optimizing for the minimization of perturbations (Imperceptibility Loss), as well as maximizing the attack efficacy (Complexity Loss), the Pareto Frontier represents the optimal boundary along which we balance these objectives. In practice, we can roughly assume that, when trained to convergence, the input perturbations will achieve a pair of objective values that lies close to this Pareto Frontier, which we verify experimentally through rigorous training and testing using the Baseline Loss defined in Eq. 6. Results are shown in Fig 1.A, where we plot values of the two optimization objectives at convergence: the Complexity objective versus the Imperceptibility objective. Each point represents the result of an evaluation of the attack on the ImageNet dataset, and different points are obtained by varying the value of γ that controls the trade-off between the two objectives. We obtain pairs of objective values that quite neatly form a boundary – which we argue to be approximately the Pareto Frontier of our two objectives.

This observed Pareto Frontier implies that, for a *given size of perturbations*, we can only expect the overall Complexity Loss to be optimized to a certain limited extent (on the Pareto Frontier) after training. If we want to further optimize the Complexity Loss past that point, it can be done by tuning γ, but doing so may come at the expense of a higher MSE (i.e., the Imperceptibility Loss). However, we would like to activate more gates, without sacrificing our MSE objective. To do this, we seek to find an optimal balance among the gating values, that pushes more gating values over the threshold τ.

To explore this idea in more detail, we elaborate using a simple example with 2 deactivated gates, and visualize the loss contours based on their gating values G_1 and G_2 in Fig. 1.B. In order to activate these gates, gating values G_1 and G_2 need to reach τ or higher, which represents the area beyond the right and top edge of the figure, respectively. When optimizing these gates using the Complexity Loss $\mathcal{L}_{\text{C}}$, we get a loss of $-\min(G_1, \tau) - \min(G_2, \tau) = -(G_1 + G_2)$. Since $-(G_1 + G_2)$ is linear, the loss contours are straight lines. In order to minimize the loss at a point (G_1, G_2), we can go in the direction of the gradient $-\nabla[-(G_1 + G_2)] = (\frac{\partial(G_1+G_2)}{\partial G_1}, \frac{\partial(G_1+G_2)}{\partial G_2})^T = (1,1)^T$, where $(1,1)^T$ is a vector indicating the direction on the loss contour map, visualized as arrows pointing to the top right at 45°.

In order to keep a good image quality and maintain a low MSE, there is a certain limit to how much we can optimize the loss $-(G_1 + G_2)$, which sets a *soft limit* on how high the sum of gating values can get. This can be represented by a "budget" B, visualized as a dotted line $-(G_1 + G_2) = B$ in Fig. 1.B. This implies that top right corner beyond the dotted line is inaccessible if we want to keep the image quality at a certain good level. Moreover, if we use the Complexity Loss, many gradient arrows will be directed towards the dotted line, where none of the gating values can reach τ. To improve upon this, we aim to change the shape of loss contours such that at least one of the gating values will reach τ. This allows us to keep perturbations to a minimum (i.e., staying within budget B), while driving the optimization to maximize the number of activated gates.

Power Loss. Based on our insights, we propose a Power Loss which can improve the shape of the loss landscape, by *pulling the contour lines outwards*, allowing us to optimize towards a spot that activates more gates. We rewrite the Complexity Loss $\mathcal{L}_{\text{C}}$ into a Power Loss $\mathcal{L}_{\text{P}}$ as:

$$\mathcal{L}_{\text{P}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = -\sum_{i=1}^{N} \min(G_i(\boldsymbol{x}_0, \boldsymbol{\delta}), \tau)^{\alpha}, \tag{7}$$

where $\alpha \in [1, +\infty)$ can be used to control the deformation of the contour line. Note that, for the case of $\alpha = 1$, the loss becomes the original Complexity Loss. For deactivated gates, the loss contour lines can be formulated as $-\min(G_1, \tau)^{\alpha} - \min(G_2, \tau)^{\alpha} = -(G_1^{\alpha} + G_2^{\alpha}) = d$. As $\alpha \geq 1$, the contour lines will now *bend outwards*. During optimization of the Power Loss, the gradient of the loss contour map can be calculated as $-\nabla[-(G_1^{\alpha} + G_2^{\alpha})] = (\frac{\partial(G_1^{\alpha}+G_2^{\alpha})}{\partial G_1}, \frac{\partial(G_1^{\alpha}+G_2^{\alpha})}{\partial G_2})^T = (\alpha G_1^{\alpha-1}, \alpha G_2^{\alpha-1})^T$, which will prioritize the optimization of higher gating values more, giving them a larger gradient magnitude and pushing them to reach τ.

We visualize the case $\alpha = 2$ and $\alpha = 3$ in Fig. 1.C, where the gradient arrows are now pointing more towards the the right or top edge, whichever is closer. This allows the depicted points in the figure to optimize a gating value to reach τ, while within budget B. We note that a larger α

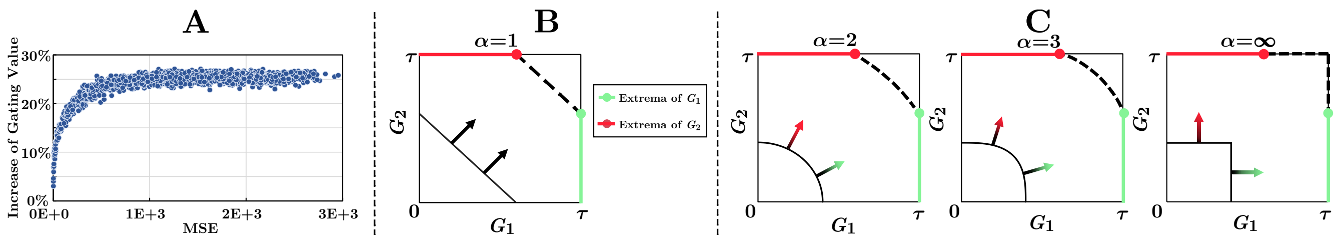

Fig. 1. A: Plot showing the trade-off between the two optimization objectives. MSE denotes the value of L_{MSE}. B: Visualization of loss contours of two gating values, where gradients are represented by arrows. C: Visualization of loss contours with different α for $\mathcal{L}_{\text{P}}$. Higher α leads to larger deformation of the contour lines, with the gradient arrows pointing away from the dotted line.

deforms the contour lines more, which makes the gradients directed more horizontally or vertically. In the extreme case, e.g., $\alpha = \infty$, the gradient direction is either vertical or horizontal. We conduct ablation experiments to find an optimal α for our method.

4.3 Rectified Direction of Gradient

In the previous section, our Power Loss balances the optimization between all the *deactivated gates*, prioritizing the loss magnitudes of some deactivated gates over others. In this section, instead of focusing on deactivated gates only, we look at how we can optimize the deactivated gates while *not interfering with the activated gates*. This aims to mitigate the difficulties in jointly optimizing many gates at once. For example, there are 32 gates in SkipNet [19] that dynamically determine which of the 32 layers to skip or not skip, and it can be difficult to jointly optimize and keep many gates activated simultaneously. In this case, the joint optimization using the Complexity Loss might lead to a scenario where, after a backpropagation step, we manage to activate some gates but cause some others to become deactivated. We attribute such optimization difficulties to the presence of conflicting gradients between the gates, which makes it hard to activate new gates without interfering with already-activated gates, resulting in a weaker attack. Below, we first explain the problem of gradient conflicts in more detail, before proposing our method to tackle it.

Gradient Conflicts can occur between the gradients meant to attack different gates. Let us represent a gradient that attacks the deactivated gates (gates with value below the threshold τ) as $\boldsymbol{g}_{\text{C}}$, and a gradient that attacks the already-activated gates (gates with values above the threshold τ) as $\boldsymbol{g}_{\text{F}}$. We call $\boldsymbol{g}_{\text{C}}$ the Complexity Gradient, and $\boldsymbol{g}_{\text{F}}$ the Finished Gradient. In the Complexity Loss defined in Eq. 6, only gradients $\boldsymbol{g}_{\text{C}}$ from gates which are deactivated are applied for parameter updating. However, as illustrated in Fig. 2.B, $\boldsymbol{g}_{\text{C}}$ might be in a largely conflicting direction from $\boldsymbol{g}_{\text{F}}$, which might result in a higher tendency for some of the activated gates to become deactivated again. This kind of disagreement will drive the gates to switch back and forth between being activated and deactivated. We show an example by focusing on the gating values of various gates of SkipNet in Fig. 2.A. Gates that have already been activated, can easily become deactivated, making it difficult to jointly activate them all at once.

Furthermore, this behaviour in the gradients cannot be simply solved by updating using both the Complexity Gradient $\boldsymbol{g}_{\text{C}}$ and the Finished Gradient $\boldsymbol{g}_{\text{F}}$. If we jointly use all gradients as in Eq. 5, we can see in Fig. 2.B that the gradient conflict might result in the joint gradient being in a very different direction from $\boldsymbol{g}_{\text{C}}$, which might obstruct the activation of other currently deactivated gates. In addition, encouraging higher activation values for those already-activated gates by including $\boldsymbol{g}_{\text{F}}$ does not further contribute to the complexity increase, yet also reduces the focus on perturbation minimization as discussed in Sec. 4.1.

Complexity Gradient Masking. As discussed, due to the conflicts in gradients, it is difficult to jointly activate many gates at once. One lingering question remains: should we update using gradients $\boldsymbol{g}_{\text{F}}$ during updating, or not? It seems like we are made to choose between 1) constantly trying to activate more deactivated gates which might lead to deactivation of already-activated gates and 2) prioritizing to keep the activated gates while focusing less on attacking deactivated gates. However, we can tackle the issue with the following insight: To prevent de-activation of activated gates, we need not update using $\boldsymbol{g}_{\text{F}}$ to further attack already-activated gates. Instead, we only need to prevent the Complexity Gradient $\boldsymbol{g}_{\text{C}}$ from *negatively affecting the already-activated gates*. To this end, we propose a Complexity Gradient Masking (CGM) technique that resolves this gradient conflict issue, such that more gates can be activated without interfering with the already-activated gates. Firstly, we compute the Finished Loss $\mathcal{L}_{\text{F}}$ for those activated gates. The formula of the Finished Loss is similar to the Complexity Loss $\mathcal{L}_{\text{C}}$ in Eq. 6, and is as follows:

$$\mathcal{L}_{\text{F}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = -\sum_{i=1}^{N} \max(G_i(\boldsymbol{x}_0, \boldsymbol{\delta}), \tau), \tag{8}$$

where as compared to Eq. 6, min is changed to max to change the focus of the loss to the activated gates. Using the Finished Loss $\mathcal{L}_{\text{F}}(\boldsymbol{x}_0, \boldsymbol{\delta})$, we can compute the Finished Gradient $\boldsymbol{g}_{\text{F}} \in \mathbb{R}^{3\times H\times W}$ as $\boldsymbol{g}_{\text{F}} = \nabla_{\boldsymbol{\delta}} \mathcal{L}_{\text{F}}(\boldsymbol{x}_0, \boldsymbol{\delta})$, where $\boldsymbol{g}_{\text{F}}$ is the direction to optimize the perturbation $\boldsymbol{\delta}$ to further increase the gate value for activated gates. Next, we can use either the Complexity Loss $\mathcal{L}_{\text{C}}$ (in Eq. 6) or the Power Loss $\mathcal{L}_{\text{P}}$ (in Eq. 7) to calculate the Complexity Gradient $\boldsymbol{g}_{\text{C}} \in \mathbb{R}^{3\times H\times W}$, which is the gradient direction to increase gating values for currently deactivated gates. If we use the Complexity Loss $\mathcal{L}_{\text{C}}$, $\boldsymbol{g}_{\text{C}}$ is computed as:

$$\boldsymbol{g}_{\text{C}} = \nabla_{\boldsymbol{\delta}} \mathcal{L}_{\text{C}}(\boldsymbol{x}_0, \boldsymbol{\delta}) = \nabla_{\boldsymbol{\delta}} \sum_{i=1}^{N} -\min(G_i(\boldsymbol{x}_0, \boldsymbol{\delta}), \tau). \tag{9}$$

Thirdly, we project the Complexity Gradient onto the Finished Gradient using:

$$\text{proj}_{\boldsymbol{g}_{\text{F}}} \boldsymbol{g}_{\text{C}} = \frac{\langle \boldsymbol{g}_{\text{C}}, \boldsymbol{g}_{\text{F}} \rangle}{\|\boldsymbol{g}_{\text{F}}\|_2} \frac{\boldsymbol{g}_{\text{F}}}{\|\boldsymbol{g}_{\text{F}}\|_2}, \tag{10}$$

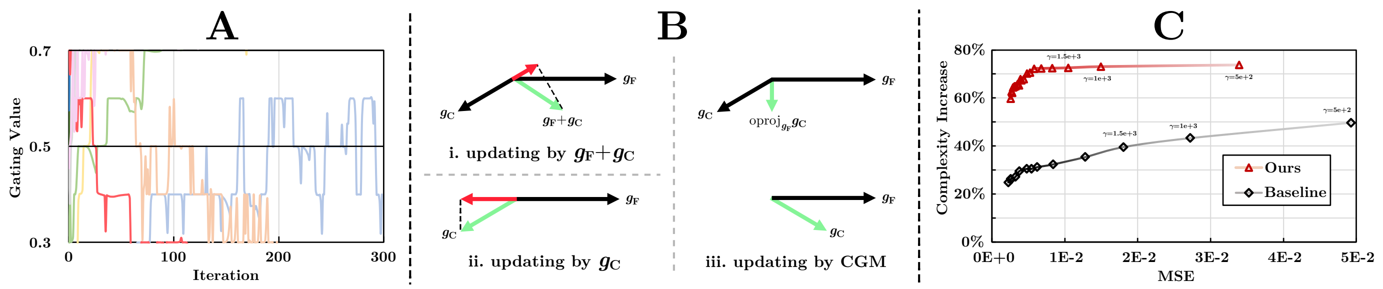

Fig. 2. A: Visualization of various gating values when optimizing using Eq. 6 to attack SkipNet. B: Illustration of gradient conflicts. (Top left) Optimizing according to Eq. 5. (Bottom Left) Optimizing according to Eq. 6. (Right) Applying our CGM according to Eq. 12. C: Visualization of the increased number of activated gates at lower MSE values using our GradMDM.

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

$$\text{oproj}_{\boldsymbol{g}_F} \boldsymbol{g}_C = \boldsymbol{g}_C - \text{proj}_{\boldsymbol{g}_F} \boldsymbol{g}_C, \tag{11}$$

whose direction is orthogonal to the Finished Gradient. Hence updating with it leads to less negative impact on the gates that have already been activated. Finally, we rectify the direction of Complexity Gradient as:

$$\boldsymbol{g}'_C = \begin{cases} \boldsymbol{g}_C, & \left\langle \text{proj}_{\boldsymbol{g}_F} \boldsymbol{g}_C, \boldsymbol{g}_F \right\rangle \geq 0, \\ \text{oproj}_{\boldsymbol{g}_F} \boldsymbol{g}_C, & \left\langle \text{proj}_{\boldsymbol{g}_F} \boldsymbol{g}_C, \boldsymbol{g}_F \right\rangle < 0 \end{cases}, \tag{12}$$

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.

Fig. 2.B visualizes the change of the angle θ between the Finished Gradient and the Complexity Gradient when our CGM method is used during training. When the angle is obtuse, the Complexity Gradient conflicts with the Finished Gradient, i.e., updating along the original Complexity Gradient has a high risk of deactivating the activated gates. Thus, our method rectifies the direction towards a direction that is orthogonal to the Finished Gradient, which mitigates the negative impact on the activated gates. When the angle is acute, the Complexity Gradient and Finished Gradient do not conflict, and the vanilla Complexity Gradient is used. Overall, this allows CGM to activate more gates and effectively avoid deactivating already-activated gates.

4.4 Training details

The techniques as described in 4.2 and 4.3 can be applied independently to optimize perturbations. When only Power Loss is applied, the perturbations are optimized using $\gamma\mathcal{L}_{\text{MSE}} + \mathcal{L}_P$. When only CGM is applied, Eqs. 8- 12 are computed, and $\boldsymbol{g}'_C + \nabla_{\boldsymbol{\delta}}\gamma\mathcal{L}_{\text{MSE}}$ is used for updating. Using the complete GradMDM method, when both techniques are used together, Eq. 8 is first computed, then Power Loss $\mathcal{L}_P$ is used to compute $\boldsymbol{g}_C$ as $\boldsymbol{g}_C = \nabla_{\boldsymbol{\delta}}\mathcal{L}_P(\boldsymbol{x}_0, \boldsymbol{\delta})$, before computing Eqs. 10-12 as usual to obtain $\boldsymbol{g}'_C$. Lastly, $\boldsymbol{g}'_C + \nabla_{\boldsymbol{\delta}}\gamma\mathcal{L}_{\text{MSE}}$ is used to update the perturbations.

Additionally, in order to balance the importance of different gates according to their corresponding computational complexities, we introduce in $\mathcal{L}_C$ (Eq. 6), $\mathcal{L}_P$ (Eq. 7) and $\mathcal{L}_F$ (Eq. 8) weights $\{\lambda_i\}_{i=1}^N$ that reweigh the losses of corresponding gates $\{G_i\}_{i=1}^N$ according to their complexities. We define λ_i as: $\lambda_i = \frac{C_i}{\sum_{j=1}^N C_j}$, where C_i denotes the computational complexity (in GFLOPs) of the i-th gate's optional operations, i.e., λ_i denotes the complexity proportion of i-th gate's operations to all the optional operations in the network.

Implementation details. We set our $\alpha = 4$, $\gamma = 100$ and run 100 training iterations for our method. The first 20% of training iterations are warm-up iterations (e.g., using Eq. 6) to get our performance closer to the Pareto Frontier, and the rest are GradMDM iterations. All the original tested dynamic networks set the activation threshold at $\tau = 0.5$, which we follow. Our code is implemented using the Pytorch framework. We use the official publicly available code for all the dynamic models. As pre-trained models are publicly available for SkipNet and SACT on ImageNet and CIFAR-10 datasets, we directly run our attacking experiments on them. For the other dynamic models and datasets, pre-trained models are not available, so we train them ourselves for each dataset by referring to the training details and hyperparameters in their respective works.

5 Experiments

We validate our approach by using it to attack popular dynamic depth networks (SkipNet [19] and SACT [20]) and a dynamic width network (ManiDP [21]) on ImageNet [50] and CIFAR-10 [51].

Dataset Settings. For evaluation, we follow the datasets and settings of previous works [22]. We evaluate on ImageNet [50] and CIFAR-10 [51], where images from ImageNet and CIFAR-10 are converted into 3×224×224 and 3×32×32, respectively.

Metrics. We follow the metrics used in previous works [22] to evaluate the effectiveness of attacks. We measure the amount of computation savings of the dynamic network that are invalidated by our attacks, and report the **Average Recovery Percentage (ARP)** of the reduced FLOPs during inference. To measure the quality of the perturbed image, we adopt the **Peak Signal-to-Noise Ratio (PSNR)** metric, which approximates the human perceptual quality and is commonly employed to evaluate image quality [52]. The PSNR can be defined as $\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}_I^2}{\text{MSE}}\right)$, where MAX_I is the maximum possible pixel value of the image. For an original m × n image I and the perturbed image K, **Mean Squared Error (MSE)** can be calculated by $\text{MSE} = \frac{1}{3HW}\left\|\boldsymbol{x}'_0 - \boldsymbol{x}_0\right\|_2^2$.

5.1 Attack on Dynamic Depth Network

We compare our approach with state-of-the-art methods ILFO [22] and DeepSloth [39]. For a fair comparison, we follow the experimental settings in [22].

Model Settings. We attack SkipNet [19], a conditional skipping network, with two different settings: **SkipNet** and **SkipNet+HRL**, where +HRL indicates that hybrid reinforcement learning is used. SkipNet+HRL achieves better efficiency compared to SkipNet and is more challenging to attack. We also attack **SACT** [20], an early termination network. Since ILFO only reported results on SkipNet and SACT, we further re-implement their code on SkipNet+HRL to compare it with our GradMDM. Note that ILFO uses a similar loss to the baseline described in Eq. 6, except that their formulation instead uses different pre-set thresholds for different gates. DeepSloth [39] is originally designed for attacking early termination-based networks only, thus we report its results on SACT only.

Results. We present the results on ImageNet in Table 1 and on CIFAR-10 in Table 11. On both CIFAR-10 and ImageNet, our GradMDM outperforms the ILFO baseline on all tested models across all metrics. This implies that our GradMDM attacks consistently achieve significantly higher increase in computational complexity, while using a less imperceptible perturbation (i.e., smaller MSE and larger PSNR). Specifically, for SkipNet and SACT, GradMDM can almost achieve 100% average recovery, which means that the dynamic networks have almost the same computational complexity as their static counterparts, which shows its effectiveness.

TABLE 1
Performance comparison of dynamic depth networks on ImageNet.

Model	SkipNet			SACT			SkipNet+HRL		
	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑
ILFO [22](baseline)	81.4	0.26	54.0	91.1	0.25	54.1	49.6	0.25	54.3
DeepSloth [39]	-	-	-	90.7	0.80	49.1	-	-	-
GradMDM(ours)	**99.3**	**0.08**	**59.1**	**99.0**	**0.10**	**57.9**	**73.4**	**0.11**	**57.5**

TABLE 2
Performance comparison of dynamic depth networks on CIFAR-10.

Model	SkipNet			SACT			SkipNet-HRL		
	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑
ILFO [22](baseline)	84.3	0.24	54.3	72.5	0.25	54.1	5.2	0.24	54.4
DeepSloth [39]	-	-	-	92.5	0.92	48.5	-	-	-
GradMDM(ours)	**99.1**	**0.11**	**57.6**	**99.0**	**0.10**	**58.0**	**30.0**	**0.11**	**57.4**

5.2 Attack on Dynamic Width Network

We next evaluate our attack on the dynamic width network ManiDP [21]. We evaluate our approach on ImageNet and CIFAR-10, with results shown in Table 3. Our GradMDM outperforms ILFO on both datasets on all metrics, showing its efficacy on dynamic width structures as well.

TABLE 3
Performance comparison of ManiDP on CIFAR-10 and ImageNet.

Dataset	CIFAR-10			ImageNet		
	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑
ILFO [22] (baseline)	80.3	0.26	54.0	85.6	0.26	54.0
GradMDM (ours)	**99.0**	**0.11**	**57.5**	**99.3**	**0.11**	**57.7**

5.3 Ablation Studies

We conduct ablation studies on ImageNet using SkipNet+HRL to further investigate GradMDM.

Components of GradMDM. Tab. 4 evaluates the gains brought by each individual component. Each individual component of our method, CGM and the Power Loss (PL), can attain better attacking performance as compared to the baselines that use Eq. 5 or Eq. 6 for optimization. Together, PL and CGM form our full method GradMDM, and perform better when used together.

Efficiency. In Tab. 5 we evaluate the time cost of performing GradMDM. Experiments are conducted on a Nvidia RTX 3090 GPU. On all models, our GradMDM conducts attacks faster than ILFO.

TABLE 4
Ablation of individual components of GradMDM. PL and CGM denote Power Loss and Complexity Gradient Masking, respectively. GradMDM uses the combination of both PL and CGM.

Method	Eq. 5	Eq. 6	CGM (Ours)	PL (Ours)	GradMDM (Ours)
Recovery (%) ↑	48.5	49.6	54.0	69.0	73.4
MSE ↓	0.25	0.25	0.11	0.11	0.11
PSNR ↑	54.3	54.3	57.5	57.6	57.5

TABLE 5
Efficiency comparison of different methods on ImageNet.

Model	SkipNet			SACT			SkipNet+HRL		
	ARP(%)	Iteration	Time (s)	ARP(%)	Iteration	Time (s)	ARP(%)	Iteration	Time (s)
ILFO [22](baseline)	81.4	300	2.1	91.1	300	1.8	49.6	300	2.0
DeepSloth [39]	-	-	-	90.7	550	3.8	-	-	-
GradMDM(ours)	**99.3**	100	**1.2**	**99.0**	100	**1.1**	**73.4**	100	**1.1**

Weight of Imperceptibility Loss γ. Tab. 6 evaluates the attacking performance and the deviation of modified input images under different settings of γ. As expected, when γ decreases, the efficacy of our attack improves, but the magnitude of the perturbation also increases. When γ is set to a larger value of $1e+4$, a very high PSNR (73.9) and low MSE (3e-3) is obtained while still simultaneously improving upon the baseline attacking performance (57.5% as compared to baseline of 49.6%). To better compare between GradMDM and ILFO, we run both methods at more γ settings and show the results in Fig. 2.C, where our method consistently attains a better Complexity-MSE trade-off.

TABLE 6
Ablation of the different settings of γ. The higher the value of γ, the more weight will be given to the Imperceptibility Loss in comparison to the Complexity Loss.

γ	1e+0	1e+1	1e+2	1e+3	1e+4
ARP (%) ↑	76.1	74.5	73.4	72.0	57.5
MSE ↓	0.25	0.21	0.11	0.03	3e-3
PSNR ↑	54.3	54.7	57.5	63.8	73.9

Classification Accuracy. We also investigate the change in image classification accuracy after our attack, and report the results for several dynamic network methods on ImageNet (corresponding to Tab. 1) in Tab. 7. We find that the perturbations by GradMDM lead to a significant drop in model accuracy. However, during our attack, we can also add a classification loss (**GradMDM w/ classification loss**), which we find helps to improve the classification accuracy while maintaining the attack performance.

TABLE 7
Classification Accuracy on ImageNet after the attack.

Setting	SkipNet		SACT		SkipNet-HRL	
	ARP(%)	Acc(%)	ARP(%)	Acc(%)	ARP(%)	Acc(%)
GradMDM	99.3	36.2	99.0	33.8	73.4	23.1
GradMDM w/ classification loss	99.3	77.5	99.0	78.2	73.2	76.9
Original Acc. (Upper Bound)	-	77.5	-	78.2	-	76.9

Power Loss hyperparameter α. In Tab. 8, we evaluate the impact of different settings of α. Different values of α all lead to improvement in performance over the baseline where $\alpha = 1$. We find that increasing α past 4 does not lead to further improvements, possibly because that will focus too much on particular larger gradients. Another way we can set hyperparameters α is by setting them individually for each gate. When we carefully choose different α values for different gates, our performance can indeed improve slightly, with ARP, MSE and PSNR of 74.5%, 0.11 and 57.5. Nevertheless, it can be difficult to tune the α hyperparameters in this manner for different models and datasets. Furthermore, the performance improvement from the tuning of α for each gate is not very significant. Thus, we report our results with constant α ($\alpha = 4$) for all gates.

TABLE 8
Ablation of the different settings of α. The higher the value of α, the larger the deformation of the contour lines in the Power Loss.

α	1	2	3	4	5	6
ARP (%) ↑	50.7	53.0	72.1	73.4	72.5	69.0
MSE ↓	0.11	0.11	0.11	0.11	0.11	0.11
PSNR ↑	57.5	57.5	57.6	57.5	57.7	57.6

Performance on samples that require different "budgets". We also conduct the following experiment on SkipNet using ImageNet: we select a subset of samples that can lead to 100% activation using the baseline (Subset A) under a controlled amount of perturbation, and a subset of samples that cannot be satisfactorily tackled by the baseline (Subset B). We find that the ARP of GradMDM on Subset A is still 100%, while on Subset B, GradMDM leads to significant improvements on ARP (80.9% vs 99.0%). This shows the efficacy of our method.

Tanh vs projection. In Eq. 2 we use the tanh formulation to map each element of our raw $(\mathbf{x}_0 + \boldsymbol{\delta})$ into the feasible space $[0, 1]$ for fair comparison with previous works [22]. Alternatively, we can also directly project $(\mathbf{x}_0 + \boldsymbol{\delta})$ into the feasible space $[0, 1]$, in a way similar to Projected Gradient Descent (PGD). Using this alternative setting, our GradMDM still outperforms the baseline on ARP metric (45.0% vs 77.5%) with a similar MSE (0.02 vs 0.02).

L2 vs L∞. In Eq. 3 we use the L_2 norm for a fair comparison against previous works [22]. Alternatively, following some other adversarial attack works [53], we can also apply the L_∞ norm. On this setting, our experiments show that GradMDM still outperforms the baseline on ARP metric (50.0% vs 73.2%) with a lower MSE (0.34 vs 0.27).

Other baselines. We also conduct experiments on two other baselines. In the **Joint (Eq. 5)** setting, we directly use Eq. 5 to optimize the perturbations, which is equivalent to directly updating using both the Complexity Gradient $\boldsymbol{g}_{\mathrm{C}}$ and the Finished Gradient $\boldsymbol{g}_{\mathrm{F}}$ in a joint manner. The other baseline is the $\mathcal{L}_P + \mathcal{L}_F$ **(Eq. 7 + Eq. 8)** setting, where we iteratively optimize the perturbation with $\mathcal{L}_P + \mathcal{L}_F$ instead of using our CGM technique. Different from our GradMDM and ILFO [22] baseline, these baselines directly update the perturbation using gradients from $\mathcal{L}_F$ during optimization, and thus tend to over-optimize the gating values of already-activated gates, and lead to sub-optimal performance.

TABLE 9
Performance comparison against other baselines.

Method	ARP(%)↑	MSE↓	PSNR↑
Joint (Eq. 5)	38.1	0.23	54.4
$\mathcal{L}_P + \mathcal{L}_F$ (Eq. 7 + Eq. 8)	66.8	0.11	57.5
GradMDM(ours)	73.4	0.11	57.5

Other dynamic architectures. To further evaluate the efficacy of our proposed GradMDM attack, we also perform experiments on other dynamic architectures, We also report results on three additional dynamic network methods: DVT [31], A-ViT [32] and Shallow-deep [33]. DVT is a Transformer architecture with dynamic depth, A-ViT is a Transformer architecture with dynamic width, and Shallow-deep is an early-exit CNN architecture with a different working mechanism from SACT. As shown in Tab. 10, GradMDM significantly outperforms the baseline on these dynamic networks as well.

TABLE 10
Attack performance comparison of other dynamic networks on ImageNet.

Model	DVT			A-ViT			Shallow-Deep 50%-Confidence		
	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑
ILFO [22](baseline)	85.0	0.26	54.1	92.7	0.11	57.6	83.7	0.29	53.6
GradMDM(ours)	**98.3**	**0.19**	**55.3**	**95.0**	**0.10**	**58.0**	**98.0**	**0.27**	**53.8**

Other datasets. We also report the performance of GradMDM on other popular image datasets: CIFAR-100 [51] and CUB-200-2011 [54] in Tab. 11 and Tab. 12 respectively. On both of these datasets, we perform attacks on several dynamic networks (SkipNet, SACT and SkipNet-HRL). Our GradMDM consistently outperforms ILFO on all settings.

TABLE 11
Attack performance comparison on CIFAR-100.

Model	SkipNet			SACT			SkipNet-HRL		
	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑
ILFO [22](baseline)	59.4	0.24	54.3	67.1	0.25	54.1	6.11	0.48	51.3
GradMDM(ours)	**98.0**	**0.03**	**63.5**	**96.9**	**0.10**	**58.3**	**24.5**	**0.14**	**56.7**

TABLE 12
Attack performance comparison on CUB-200-2011.

Model	SkipNet			SACT			SkipNet-HRL		
	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑	ARP(%)↑	MSE↓	PSNR↑
ILFO [22](baseline)	94.4	0.10	58.4	94.7	0.09	58.5	57.2	0.23	54.6
GradMDM(ours)	**95.9**	**0.09**	**58.7**	**96.0**	**0.08**	**59.0**	**76.6**	**0.15**	**56.5**

Transferability of Perturbations. We also investigate the transferability of perturbations generated by GradMDM between different models as shown in Tab. 13. Specifically, we evaluate two scenarios: 1) where the perturbations from SkipNet are used to attack SkipNet-HRL, and vice versa; 2) where the perturbations from SkipNet are used to attack ManiDP. Results show that the perturbations from GradMDM are transferable to some extent, which opens up the potential of it being used as a black-box attack.

TABLE 13
Evaluation of the transferability of GradMDM perturbations between models on ImageNet.

SkipNet→SkipNet-HRL			SkipNet-HRL→SkipNet			SkipNet→ManiDP		
ARP(%)	MSE	PSNR	ARP(%)	MSE	PSNR	ARP(%)	MSE	PSNR
42.5	0.11	57.6	53.1	0.11	57.4	93.7	0.11	57.6

6 Discussion of Defense Measures

In this section, we discuss some potential defensive measures against our GradMDM attack. A possible countermeasure is to perform adversarial training [55] to improve adversarial robustness, which is an approach that has been empirically shown to be effective against many accuracy-based attacks on static networks. Furthermore, some other simple *out-of-the-box defense* approaches such as Spatial Smoothing [56] and JPEG Compression [57] can also be employed to defend against our proposed attack by pre-processing each input image. Here, we evaluate GradMDM against several defense measures and report the results in Tab. 14. We find that these defense measures are beneficial to improve robustness.

7 Applications and Significance

The aim of energy-oriented attacks on dynamic networks is to increase their computation complexity, which leads

TABLE 14
GradMDM attack performance comparison on ImageNet under various defense measures.

Method	ARP(%)
Adversarial Training [a]	39.2
Spatial Smoothing [b]	49.0
JPEG Compression [c]	43.8
No defense measures	73.4

to increased latency in response. Thus, in practical scenarios, attacks against dynamic networks can be analogous to denial-of-service attacks [58]. This can be dangerous in some safety-critical applications such as self-driving vehicles, where a timely response is important [59]. Such attacks are also disruptive in some cloud-based Internet-of-Things (IoT) applications [39], where an optional model partition is deployed on the cloud, and network latency can be increased significantly by forcefully activating the model's computations on the cloud, leading to excessive transmissions between cloud servers and the IoT device. Lastly, the increased energy consumption from the extra computations can cause devices to run out of battery faster, which can be exploited by malicious parties to disable these devices (e.g., security robots, mobile devices) [60].

8 Conclusion

In this work, we introduce GradMDM, a novel energy-oriented attack that adjusts the magnitude (using a Power Loss) and direction (using CGM) of gradients to effectively find a small perturbation that activates more computation units of dynamic networks during inference. When tested on several datasets and dynamic networks, GradMDM consistently outperforms existing works on all metrics.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[2] Gao Huang, Zhuang Liu, et al. Densely connected convolutional networks. In *CVPR*, 2017.
[3] Alexey Dosovitskiy, Lucas Beyer, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.
[4] Ze Liu, Yutong Lin, Yue Cao, et al. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.
[5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, et al. A convnet for the 2020s. *CVPR*, 2022.
[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
[7] Olaf Ronneberger et al. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*. Springer, 2015.
[8] Vijay Badrinarayanan et al. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017.
[9] Liang-Chieh Chen and George others Papandreou. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017.
[10] Kaiming He, Georgia Gkioxari, et al. Mask r-cnn. In *ICCV*, 2017.
[11] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.
[12] Wei Liu, Dragomir Anguelov, et al. Ssd: Single shot multibox detector. In *ECCV*. Springer, 2016.
[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.
[14] Nicolas Carion, Francisco Massa, et al. End-to-end object detection with transformers. In *ECCV*. Springer, 2020.
[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. Attention is all you need. *NeurIPS*, 30, 2017.
[16] Yu Cheng et al. A survey of model compression and acceleration for deep neural networks. *arXiv*, 2017.
[17] Jian Cheng et al. Recent advances in efficient computation of deep convolutional neural networks. *FITEE*, 19(1):64–77, 2018.
[18] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *TPAMI*, 2021.
[19] Xin Wang, Fisher Yu, Zi-Yi Dou, et al. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018.
[20] Michael Figurnov, Maxwell D Collins, et al. Spatially adaptive computation time for residual networks. In *CVPR*, 2017.
[21] Yehui Tang, Yunhe Wang, et al. Manifold regularized dynamic network pruning. In *CVPR*, 2021.
[22] Mirazul Haque, Anki Chauhan, Cong Liu, and Wei Yang. Ilfo: Adversarial attack on adaptive neural networks. In *CVPR*, 2020.
[23] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018.
[24] Brandon Yang et al. Condconv: Conditionally parameterized convolutions for efficient inference. *NeurIPS*, 32, 2019.
[25] Yanwei Li, Lin Song, et al. Learning dynamic routing for semantic segmentation. In *CVPR*, 2020.
[26] Gao Huang, Danlu Chen, Tianhong Li, et al. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018.
[27] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv*, 2016.
[28] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *AAAI*, volume 32, 2018.
[29] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, 2018.
[30] Hengduo Li et al. 2d or not 2d? adaptive 3d convolution selection for efficient video recognition. In *CVPR*, 2021.
[31] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *NeurIPS*, 34:11960–11973, 2021.
[32] Hongxu Yin et al. A-vit: Adaptive tokens for efficient vision transformer. In *CVPR*, pages 10809–10818, 2022.
[33] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *ICML*, pages 3301–3310. PMLR, 2019.
[34] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *NeurIPS*, 32, 2019.
[35] Yoshua Bengio et al. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv*.
[36] Kyunghyun Cho and Yoshua Bengio. Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. *arXiv*, 2014.
[37] Emmanuel Bengio, Pierre-Luc Bacon, et al. Conditional computation in neural networks for faster models. *arXiv*, 2015.
[38] Ji Lin et al. Runtime neural pruning. *NeurIPS*, 30, 2017.
[39] Sanghyun Hong et al. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. In *ICLR*, 2021.
[40] Jh Pan et al. Gradauto: Energy-oriented attack on dynamic neural networks. In *ECCV*, pages 637–653. Springer, 2022.
[41] Sam Leroux et al. Iamnn: Iterative and adaptive mobile neural network for efficient image classification. *arXiv*, 2018.
[42] Qiushan Guo, Zhipeng Yu, Yichao Wu, Ding Liang, Haoyu Qin, and Junjie Yan. Dynamic recursive neural network. In *CVPR*, 2019.
[43] Haichao Yu, Haoxiang Li, Honghui Shi, et al. Any-precision deep neural networks. *arXiv*.
[44] Qing Jin, Linjie Yang, and Zhenyu Liao. Adabits: Neural network quantization with adaptive bit-widths. In *CVPR*, 2020.
[45] Jianghao Shen, Yue Wang, et al. Fractional skipping: Towards finer-grained dynamic cnn inference. In *AAAI*, volume 34, 2020.
[46] Changlin Li, Guangrun Wang, Bing Wang, et al. Dynamic slimmable network. In *CVPR*, 2021.
[47] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *S&P*. IEEE, 2017.
[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, et al. Intriguing properties of neural networks. *arXiv*, 2013.
[49] Ali Jahan, Kevin L Edwards, and Marjan Bahraminasab. *Multi-criteria decision analysis for supporting the selection of engineering materials in product design*. Butterworth-Heinemann, 2016.
[50] Jia Deng, Wei Dong, Richard Socher, et al. Imagenet: A large-scale hierarchical image database. In *CVPR*. IEEE, 2009.
[51] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
[52] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *ICPR*. IEEE, 2010.
[53] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv*, 2014.
[54] Catherine Wah et al. The caltech-ucsd birds-200-2011 dataset. 2011.
[55] Aleksander Madry et al. Towards deep learning models resistant to adversarial attacks. *arXiv*, 2017.
[56] Weilin Xu et al. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv*, 2017.
[57] Nilaksh Das et al. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv*, 2017.
[58] Francesco Palmieri et al. Energy-oriented denial of service attacks: an emerging menace for large cloud infrastructures. *J. Supercomput.*, 2015.
[59] Chuang Hu, Wei Bao, et al. Dynamic adaptive dnn surgery for inference acceleration on the edge. In *INFOCOM*, 2019.
[60] Thomas Martin et al. Denial-of-service attacks on battery-powered mobile computers. In *PerCom*. IEEE, 2004.