ILFO: Adversarial Attack on Adaptive Neural Networks

Mirazul Haque Anki Chauhan Cong Liu Wei Yang
The University of Texas at Dallas
{mirazul.haque, axc170043, cong, wei.yang}@utdallas.edu

Abstract

With the increasing number of layers and parameters in neural networks, the energy consumption of neural networks has become a great concern to society, especially to users of handheld or embedded devices. In this paper, we investigate the robustness of neural networks against energy-oriented attacks. Specifically, we propose ILFO (Intermediate Output-Based Loss Function Optimization) attack against a common type of energy-saving neural networks, Adaptive Neural Networks (AdNN). AdNNs save energy consumption by dynamically deactivating part of its model based on the need of the inputs. ILFO leverages intermediate output as a proxy to infer the relation between input and its corresponding energy consumption. ILFO has shown an increase up to 100 % of the FLOPs (floating-point operations per second) reduced by AdNNs with minimum noise added to input images. To our knowledge, this is the first attempt to attack the energy consumption of an AdNN.

1. Introduction

Deep neural networks (DNNs) have enabled impressive accuracy improvements over traditional machine learning techniques in a variety of tasks, such as object and speech recognition, and machine translation. However, these accuracy improvements depend on the availability of powerful computational resources that necessitate substantial energy consumption [25]. The computation required for deep learning research doubles every few months, resulting in an estimated 300,000x increase from 2012 to 2018 [23]. The ResNet152 architecture [12] with 152 layers, increases the accuracy by 4.4% over GoogLeNet[26] on the ImageNet dataset but the test-time becomes 14 times slower. The increasing test-time cost has become a problem, especially in cases where faster inference is required (*e.g.* mobile apps). For example, a visually impaired user may use an app based on DNNs for object detection. The prolonged response time of DNNs will cause fatal issues for the user relying on the app.

To address these issues, extensive research has studied and proposed models called *energy-saving models*. There are two types of energy-saving models: On-Device Neural Networks (ODNNs) and Adaptive Neural Networks (AdNNs). ODNN models [16, 15, 30] use lower dimensions of filters or change the dimension of input to the filters to decrease the number of computations, while AdNNs dynamically deactivate a certain part of the model based on different inputs to reduce the number of computations.

Due to aforementioned damaging consequences, attackers are motivated to attack the energy-saving models. For example, SkipNet [29], an AdNN model, can reduce 50 % of floating-point operations in a ResNet model for the CIFAR-10 dataset. If an attack on the input can increase 35 - 40 % of the reduced flops, the purpose of the SkipNet model would not be served. This type of decrease in performance will eventually jeopardize the reputation of the app or software, which uses the energy-saving model.

Investigating the robustness of these energy-saving models in terms of energy consumption is needed. The robustness of a machine learning model can be defined as the stability of the model accuracy after adding some noise to input data, *i.e.,* a robust model's accuracy will not deteriorate against noisy inputs. We define a new term *energy robustness* as the stability of the model's energy consumption after getting a perturbed input. Traditional adversarial machine learning techniques formulate their attacks based on the relation between input and output. However, energy robustness considers the relation between input and its corresponding energy consumption. In Section 4.1, we show that the attack formulation for energy robustness requires detailed knowledge about the model and needs to define the target values for different AdNNs. Before this work, there has not been any technique that can investigate the energy-robustness of a model. In this paper, we are motivated to investigate the robustness of the energy-saving models by generating perturbed images to increase the energy consumption of these models. Specifically, we use AdNN as an example to perform such investigation. In our knowledge, this is the first attempt in this direction.

However, attacking a model with respect to energy consumption is complicated because it is impossible to find a trail from input to energy consumption during inference. There are multiple systems (*e.g.*, Nvidia TX2 server, FogNode, Raspberry Pi), which can measure the energy consumption of machine learning model inference. Although the energy consumption measured using these systems is not dependent on only the model. The measurement is also dependent on the system environment. This also makes it impossible to find out the pattern in energy consumption in different images. If an image inference consumes less amount of energy, then it is difficult to conclude if it is because of the input or the system environment.

Due to the difficulty of relating input and energy consumption, we attack the energy-saving models by increasing the number of computations during inference. For traditional DNNs and ODNNs, the number of computations during inference is constant *i.e.,* we can not relate the number of computations with input. Although AdNNs change the number of computations based on intermediate layer-wise outputs. Creating a relation between input and intermediate outputs relates input and the number of computations. In this paper, we have explored the relation between inputs and intermediate outputs of AdNNs and attack these models.

We are proposing ILFO (**I**ntermediate Output-Based **L**oss **F**unction **O**ptimization) to attack AdNNs. Using the gradient between intermediate outputs and inputs, we create a loss function and optimize that function. Function optimization changes the intermediate outputs and the changed intermediate outputs activate the deactivated part of the AdNN model, increasing the number of computations. In this paper, after giving a brief idea about the different AdNN models (Section 3), we have formulated the idea of ILFO (Section 4.1). In Section 4.2, we have explained the approach ILFO. We have evaluated ILFO based on three characteristics - effectiveness, quality, and efficiency. Our results show that ILFO could increase the reduced Floating-Point Operations (FLOPs) by AdNNs during inference of the input up to 100 %.

Our paper makes the following contributions:

(i) This is the first work investigating the robustness of neural networks in terms of energy consumption.

(ii) ILFO could increase the reduced FLOPs by AdNNs during inference up to 100 %.

(iii) ILFO tries to formulate a generic approach towards attacking all types of AdNNs.

2. Related Works

AdNNs. There has been an extensive amount of research in the field of decreasing energy consumption during the test-time of CNNs. AdNNs are part of that research. Before proposing AdNNs, many of the earlier works use models [6, 11, 13, 20, 14] to transfer knowledge to shallower networks. These techniques are not used in training but as the post-processing. Although these networks are not dynamic with respect to the changing inputs. Recently, there has been a number of AdNN models proposed. A few versions of the AdNNs have modified the traditional AdNN model to save energy. Graves *et al.* [9] explore the halts in Recurrent Neural Networks while Figurnov *et al.* [7] and Teerapittayanon *et al.* [28] propose the use of early termination in CNNs. Recently, Wang *et al.* [29] propose SkipNet that uses gates between the layers to skip a particular layer. Few of the AdNNs are constructed by cascading multiple networks through a simple computation unit. Bolukbasi *et al.*[2] train a termination policy for cascades of pre-trained CNNs arranged in order of increasing costs. Guan *et al.* [10] use reinforcement learning to choose between multiple classifiers. Nan *et al.* [22] use gating techniques to choose a high-cost or low-cost model.

Adversarial Examples. Adversarial Examples are the input that is fed to machine learning models to change the prediction of the model. In earlier works [4, 21, 5], 'good word attacks' or spelling modifications have long been used to bypass the spam filters. More recently, Szegedy *et al.* [27] and Goodfellow *et al.* [8] propose adversarial attacks on deep computer vision models. Karmon *et al.* [19] propose a technique to attack CNNs in which a localized patch is introduced in an image instead of adding noise to the full image. With a similar approach, adversarial attacks have been extended to various fields like text and speech processing [3, 18], and graph models [31, 1]. Although, all these attacks focus on changing the prediction and do not concentrate on increasing test-time. Our approach is the first to explore that direction.

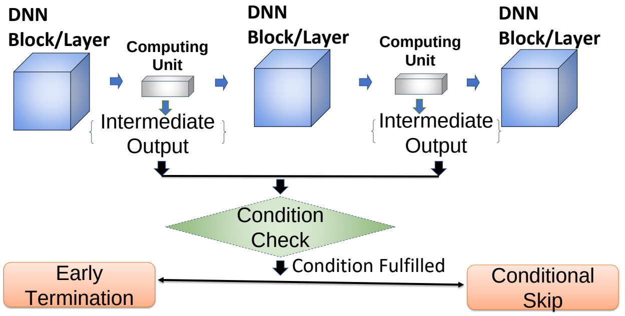


Figure 1. Working mechanism of an AdNN

3. Background

In this paper, we focus on attacking AdNNs. Figure 1 represents the working mechanism of an AdNN. The main goal of this type of network is to refrain from executing a few layers in a DNN. The AdNNs generate intermediate outputs before each layer or block. To calculate these outputs, generally, a shallow computing unit (simple CNN, RNN, or linear classifier) is added between two layers or blocks, as shown in the Figure 1. These intermediate outputs are evaluated against predefined conditions and if the output reaches a certain threshold, the condition is fulfilled. Fulfilling the condition can result in two ways. First, AdNN determines that certain layers or blocks are not required for the inference, and those layers or blocks are skipped. That type of AdNN is called *Conditional-skipping AdNN* [29]. Second, AdNN terminates the operations within a block or network early by deciding that later part of the operations are not required. We call those AdNNs as *Early-termination AdNNs* [7, 2].

The two AdNNs we have attacked are SkipNet [29] and SACT [7]. Both models are created by modifying ResNet model architecture. In the following Section 4.1, we will discuss the working mechanism of these two models. SkipNet uses gate-based computing units to selectively choose residual blocks. To decide if a block will be skipped or not, SkipNet relies on the gate output corresponding to the block. SACT examines each position of the block input and decides if that position is needed to be processed in the next layer within the block. Halting scores are calculated for each position of the input after each layer for this purpose.

4. ILFO

4.1. Problem Formulation

Our objective is to attack AdNNs by creating modified inputs which will increase the number of computations of AdNNs during inference. The attack should concentrate on two factors. First, the attacked image should increase the number of computations during inference. Second, the attacked image and original image can not be differentiated. For every AdNN, we define two states of the intermediate outputs - desirable state (des) and current state (cur). State des defines the state of the intermediate outputs for which computations during inference will be maximum. State cur is the current state of the intermediate outputs. For an input x, we search for the perturbation δ for which $f(x+\delta)$ is minimum. $f(x+\delta)$ is a loss function which represents the difference between des and cur. The optimization function can be represented as,

$$minimize(\delta + c \cdot f(x+\delta)) \text{ such that, } (x+\delta) \in [0,1]^n \tag{1}$$

where c is a suitably chosen positive constant. Our approach formulates the attack for both types of AdNNs and converts the attack in the form of equation (1).

4.1.1 Attacking Early-termination AdNN

Figure 2 represents the working mechanism of a residual block in SACT. SACT calculates halting scores as the intermediate outputs. Within a residual block, these halting scores are calculated for each position after each layer. In SACT, the condition is the cumulative halting score for a position within a block should be less than 1 to make the position active. In Figure 2, Deep green positions are active while light green positions are inactive. To increase the energy consumption in SACT, the principle approach should be minimizing the halting scores generated at each layer for each position. In that way, the position will be active for more number of layers. Formulating an attack based on the halting score would require a large number of intermediate outputs upon which the computations will depend. [1] Because of this reason, we would use the ponder cost of each position to formulate the attack. Ponder cost (ρ) of a position is the summation of the number of residual units(layers) for which the position is active and the remainder value of the cumulative halting score after exceeding 1. The ponder cost map will store the ponder costs of all input positions for a residual block. In Figure 2, P0, P1, P2, P3 are four spatial positions, ρ_0, ρ_1, ρ_2, ρ_3 are the ponder costs of those four positions. The value of ρ_0 will be 1 because P0 is active for only one layer, and there is no remainder of the cumulative halting score after exceeding 1. Similarly for ρ_2, the value will be 3.1 because P2 is active for 3 layers and the remainder is 0.1 (0.2+0.1+0.8-1.0=0.1). ρ_1 and ρ_3 will be 1.0 and 2.1 respectively. Increasing the ponder cost of each position will increase the computation of each position.

Creating an attack using the ponder cost maps can be complex because the ponder cost maps are multi-dimensional arrays, which the dimensions can be different for depending on residual blocks. Using resizing and addition, we create a smaller size 2D array to represent the ponder costs. As shown in Figure 3, ρ_{arr} represents the generated array and $\rho_{arr}[i][j]$ represents the ponder cost in the i^{th} row and j^{th} column of the ρ_{arr}. To represent our formulated attack using (1), we need to define the ideal behavior of ρ_{arr}. We want each ponder cost of the ρ_{arr} to get increased and reach a certain predefined threshold represented by ρ_T. With each iteration, Perturbation can be added to the input such that, each ponder cost in ρ_{arr} exceeds ρ_T at the end of the iterations. Before inserting perturbation at an iteration, if $\rho_{arr}[i][j]$ has exceeded ρ_T, then the value at $\rho_{arr}[i][j]$ should remain the same after adding perturbation. Otherwise, the value at $\rho_{arr}[i][j]$ should increase. We represent the principle for each iteration in the following way.

[1] For each position of each layer input, halting scores are calculated.

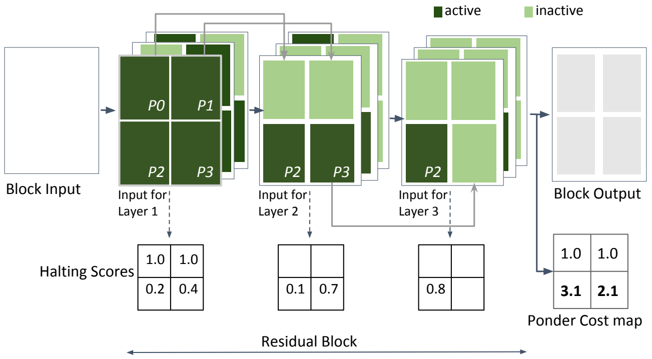


Figure 2. Working mechanism of a residual block in SACT.

$$\rho_{arr}[i][j](x+\delta'+\Delta) = \begin{cases} \rho_{arr}[i][j](x+\delta'), \text{if} \\ \rho_{arr}[i][j](x+\delta') >= \rho_T \\ \rho_{arr}[i][j](x+\delta') + \Delta\rho, \text{otherwise} \end{cases} \tag{2}$$

Here, δ' represents accumulative perturbation before the current iteration, Δ is the perturbation being added at the current iteration, x is input, $\Delta\rho$ is the value added at position $\rho_{arr}[i][j]$. Figure 3 represents the approach through example.

We can represent this maximization problem in equation (2) in the form of equation (1). For this problem, the current state (cur) can be represented by ρ_{arr}. For desirable state (des), we define another array ρ_{max} with the same dimension as ρ_{arr}, where, $\rho_{max}[i][j] = \rho_T$ for $i = 0, 1, 2....R$ and $j = 0, 1, 2....C$. R and C represent the dimension of ρ_{max}. Minimizing the difference between des and cur would increase the number of computations. We represent loss function $f(x + \delta)$ as,

$$f(x+\delta) = \sum_{i=0}^{R}\sum_{j=0}^{C} max(\rho_{max}[i][j] - \rho_{arr}[i][j], 0)$$

The max function ensures that any value in ρ_{arr} does not get decreased. We can represent the formulation problem using equation (1) replacing $f(x + \delta)$.

4.1.2 Attacking Conditional-skipping AdNN

Figure 4 represents the working mechanism of a SkipNet model containing $N + 1$ residual blocks and gates corresponding to the blocks. A block becomes active if the corresponding gate output exceeds threshold G_T. In Figure 4, the gates are represented as G_i where $i \in [0, N]$. $\alpha_i(x)$ is a boolean variable which becomes true when $Output(G_i) >= G_T$, where x is the input and G_T is the threshold value to make a gate active. $\alpha_i(x) = Output(G_i) >= G_T$

In the model shown in Figure 4, $\alpha_0(x)$ and $\alpha_N(x)$ are $True$ while $\alpha_1(x)$ is $False$. All the residual blocks will be active if $\alpha_0(x) \wedge \alpha_1(x) ..... \wedge \alpha_N(x)$ is true. If the input image is x and attacked image is $x + \delta$, then we can formulate the attack in two ways. The first way is to represent it as an optimization problem with constraints.

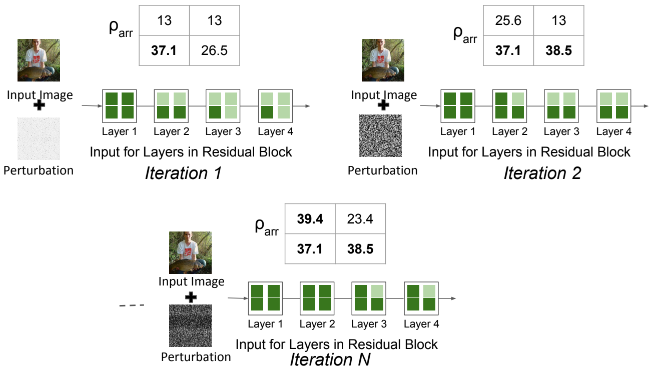


Figure 3. Attack mechanism for SACT. ρ_{arr} values less than ρ_T (37 in the example) try to exceed ρ_T after each iteration. Values greater than ρ_T try to remain the same.

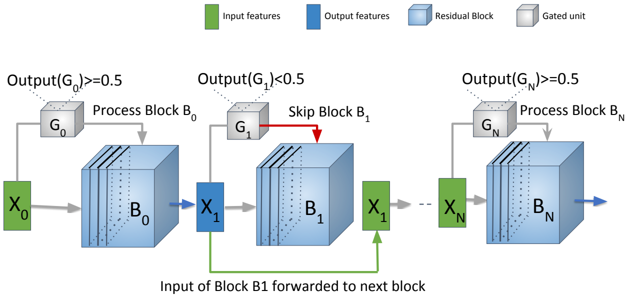


Figure 4. Working mechanism of SkipNet. $B_0, B_1..., B_N$ represents residual blocks.

$$minimize(\delta), \tag{3}$$

such that, $\alpha_0(x + \delta) \wedge \alpha_1(x + \delta)..... \wedge \alpha_N(x + \delta) = True$

Although equation (3) can fulfill our objective, nonlinearity of the constraints can increase the complexity. The other way is to represent the attack using equation (1). We define G_{arr}, a 1D array representing the gate outputs. $G_{arr}[i]$ represents i^{th} gate output. In an iterative way, we add perturbation to the input x such that the gate outputs, which have a value lower than G_T, reach G_T, and other outputs remain the same. The representation of the principle is,

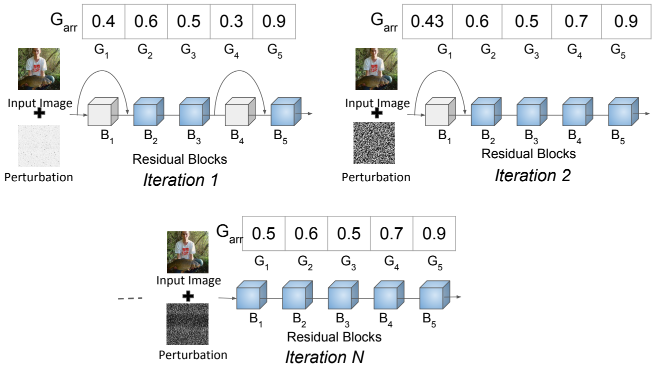


Figure 5. Attack mechanism for SkipNet. G_{arr} values less than G_T try to exceed G_T after each iteration. Values greater than G_T try to remain the same. G_T is 0.5 in this example.

$$G_{arr}[i](x+\delta'+\Delta) = \begin{cases} G_{arr}[i](x+\delta'), \text{if } G_{arr}[i](x+\delta') \\ >= G_T \\ G_{arr}[i](x+\delta') + \Delta g, \text{otherwise} \end{cases} \tag{4}$$

Here, δ' is the accumulative perturbation before the current iteration, Δ is the perturbation being added at the current iteration, Δg is the value added at position $G_{arr}[i]$. Figure 5 shows the attack process by an example.

To formulate the attack in the form of equation (1), we define $G_{arr}[i]$ as current state (cur). We define array G_{max} with same dimension as G_{arr}. G_{max} can be represented as $G_{max}[i] = G_T$ for $i = 0, 1, 2....N$ and we define G_{max} as des. We define loss function $f(x + \delta)$ as,

$f(x + \delta) = \sum_{i=0}^{N} max(0, G_{max}[i] - G_{arr}[i])$

By replacing loss function, we formulate the attack using equation (1).

4.2. Algorithm

We propose ILFO based on the equation (1). ILFO is an iterative technique that generates the best possible perturbation for which the loss function value (mentioned in 4.1) will be minimum. ILFO needs a pre-defining stage before the procedure, which helps to formulate the loss function. Figure 6 represents the pre-defining stage and the overview of ILFO. Sub-figure (a) represents the pre-defining stage. In this stage, desirable (des) and current (cur) states are defined based on the intermediate outputs of AdNN, and a single or multi-dimensional array is used to represent the states. The loss function is defined based on the difference between these two states. Sub-figure (b) represents an overview of ILFO. ILFO takes an image as input and generates the perturbed image using optimizing the loss function, which was defined in the pre-defining stage.

After adding perturbation to the image, it is important to make sure that the generated values are within the maximum and minimum limit for the inference. We use $tanh$ function for such purpose. The $tanh$ function can convert a value to a new value within the range -1 and 1. We use this property of $tanh$ function and send the distorted image values to a $tanh$ function. The optimization function for ILFO is a modified version of equation (1).

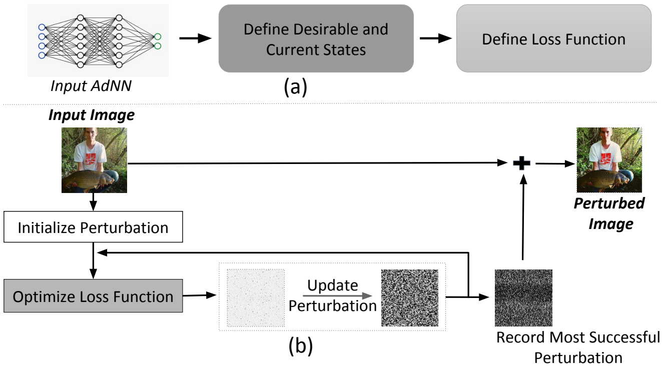


Figure 6. (a) Pre-defining the loss function. (b) Overview of ILFO.

$$\begin{aligned} minimize||\frac{1}{2} \cdot (tanh(w) + 1) - I||_2^2 + \\ c \cdot f(\frac{1}{2} \cdot (tanh(w) + 1)) \end{aligned} \tag{5}$$

Where, w is the generated perturbation and I is the input.

Algorithm 1: ILFO Technique

```
Inputs  : $I : Input\ Image$
Outputs: $F : Final\ Perturbed\ Image$
begin
    $Initialize\ \delta, L\_min, \delta\_best, des$
    $T = number\_of\_iterations$
    $iter\_no = 0$
    while $iter\_no < T$ do
        $I' = scale(\delta + I)$
        $cur = intermediate\_outputs(I')$
        $L\_dist = distance(I', I)$
        $L\_compute = loss(des, cur)$
        $L = L\_dist + c \cdot L\_compute$
        $L\_new, \delta\_new = optimizer(L, \delta)$
        $\delta = \delta\_new$
        if $L\_new < L\_min$ then
            $\delta\_best = \delta\_new$
            $L\_min = L\_new$
        end
        $iter\_no = iter\_no + 1$
    end
    $F = \delta\_best + I$
end
```

Steps mentioned in Figure 6 (b) can be explained through Algorithm 1. The algorithm takes an image to perturb as input (I) and returns the perturbed image (F).

Initializing Perturbation: This step is used for initializing variables. We initialize perturbation (δ), minimum loss recorded (L_min), most successful perturbation (δ_best) and the desired state (des) (Line 2). The desired state is formulated manually using steps of 6 (a). Number of iterations (T) is also initialized (Line 3). This step is performed only once in the algorithm.

Optimizing Loss Function: This is a step which gets executed for each iteration. A perturbed image (I') is generated by adding perturbation (δ) to the input image (I) (Line 6). $scale$ method re-scales the I' using $tanh$ function. Current output state (cur) is generated from $intermediate\_outputs$ method using inference of I' (Line 7). Two types of loss function values are generated. First loss function (L_dist) measures Euclidean distance between input image (I) and perturbed image (I') (Line 8). Second loss function ($L\_compute$) measures distance between current (cur) and desirable (des) intermediate output states using $loss$ method (Line 9). $loss$ method is defined during last step of Figure 6 (a). The main loss function (L) is created using both L_dist and $L\_compute$ (Line 10). The main loss function is optimized and updated perturbation (δ_new) and optimized loss function (L_new) is generated (Line 11).

Update Perturbation and Record Most Successful Perturbation: Perturbation variable (δ) is updated using δ_new (Line 12). If updated loss (L_new) is lesser than the recorded minimum loss (L_min) (Line 13), L_min variable is updated (Line 15) and the current solution is recorded as most successful perturbation (δ_best) (Line 14). When the iterations are finished, F is returned as the perturbed output image by adding the most successful perturbation (δ_best) with input image (I) (Line 19).

5. Evaluation

We investigate the three characteristics of our attacks in this Section: **Effectiveness** (Increase of computation after attacks), **Quality** (Image feature preservation after attacks) and **Efficiency**(Cost comparison of attacks).

5.1. Baseline Technique

We use Constraint-based technique as one of the baseline techniques. Constraint-based technique is designed based on the equation (3) and it uses an optimization technique with non-linear constraints. We only add perturbation to the border area pixels in this technique. We use this algorithm to attack only SkipNet because attacking SACT in this technique would be resource consuming. Before this attack, we define the minimum number of gates that must be activated (G_{Max}) using the attack. The constraint can be defined as $N_G >= G_{Max}$ where N_G is the number of current active gates. The detailed implementation of this technique is mentioned in Appendix A.

Table 1. Different parameters used for Constraint-based technique.

	SkipNet	
Params	CIFAR-10	ImageNet
G_{Max}	42	29
Iterations	300	10
Tolerance	1e-06	1e-06

Table 2. Different parameters used for ILFO.

	SkipNet		SACT	
Params	CIFAR-10	ImageNet	CIFAR-10	ImageNet
Iteration	1,000	300	1,000	300
Constant	100,000	100,000	100,000	100,000
learning rate	0.01	0.01	0.01	0.01

Table 3. Different selection parameters used for AdNNs.

	SkipNet		SACT	
Threshold	CIFAR-10	ImageNet	CIFAR-10	ImageNet
ρ Threshold for selection	–	–	11.5	25.0
Active gate threshold for selection	36	24	–	–

5.2. Experimental Setup

Datasets. We evaluated our techniques on the CIFAR-10 and ImageNet datasets for SkipNet and SACT models. The reason for selecting the specific datasets was to maintain consistency as both the models have been previously evaluated on the same datasets.

Models. SkipNet and SACT models can be built using ResNet models. SkipNet uses a modified ResNet-110 model to classify the CIFAR-10 images and a modified ResNet-101 model to classify the ImageNet images. SkipNet models are trained using supervised learning. SACT uses a modified ResNet-32 model to classify the CIFAR-10 images and a modified ResNet-101 model to classify the ImageNet dataset.

Table 1 and Table 2 represent the parameters used by Constraint-based technique and ILFO respectively. Tolerance is the acceptable relative error in satisfying constraint. The number of iterations for the Constraint-based technique is 10 for thr ImageNet data and 300 for the CIFAR-10 data. Due to the higher time consumption of Constraint-based technique, we have used a less number of iterations. Due to lower cost of ILFO attack, we can use 300 iterations for the ImageNet and 1,000 iterations for the CIFAR-10 dataset. The constant (c) used here is static and kept high because we are more concerned about $f(x + \delta)$ than δ (equation (1)).

5.3. Experimental Results

5.3.1 Effectiveness

We measure the effectiveness of the attacks by measuring the total increase in FLOPs [2] and increase in AdNN reduced FLOPs during the inference of attacked images. For this purpose, we process all the images from the CIFAR-10 dataset and choose the images with lower inference FLOPs

[2] For total FLOPs calculation, we calculate FLOPs for each convolutional layer [24]

count. Similarly, for the ImageNet dataset, we choose 700 images with lower inference FLOPs count. For both the models (SACT and SkipNet), we set a threshold to know if the inference FLOPs count of the image is low or not. For SkipNet, this threshold is dependent on the number of active gates, while for SACT, the threshold relies on ponder cost (ρ). The threshold values can be found in Table 3. As part of pre-processing, ImageNet images are converted into shape, $224 \times 224 \times 3$. We compare ILFO with the other two baseline approaches (Gaussian noise and Constraint-based) to measure effectiveness. We have used Constraint-based attacks only on conditional-skipping AdNN (SkipNet) because creating Constraint-based technique for early-termination AdNN (SACT) is time and resource consuming. Figure 7 represents the percentage of the total increase in FLOPs after attacks. ILFO has outperformed Constraint-based technique and could increase the number of FLOPs by 28 - 45 % in all the scenarios.

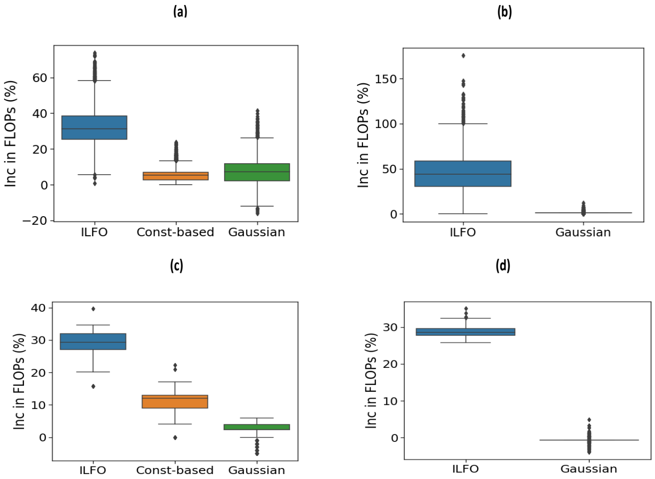


Figure 7. Percentage of FLOPs increased by two attacks. (a) Results after attacking SkipNet using CIFAR-10 images. (b) Results after attacking SACT using CIFAR-10 images. (c) Results after attacking SkipNet using ImageNet images. (d) Results after attacking SACT using ImageNet images.

The effectiveness of an attack for AdNN can also be shown by the increase in the percentage of reduced FLOPs during inference. AdNN decreases the number of computations of a traditional DNN during inference. We measure the percentage of the reduced FLOPs that is increased using ILFO and show the results in Figure 13 (in Appendix) and Table 4. ILFO has been able to increase up to 100 % of the reduced FLOPs in all four scenarios (For two models and two datasets). The average percentage of increasing the reduced FLOPs is between 72 - 91 %. These results also explain that not having greater scope to increase the FLOPs is the reason why ILFO could only increase 30 - 35 % of FLOPs on average for the ImageNet dataset.

Table 4. Average increase of reduced AdNN FLOPs by ILFO.

	SkipNet		SACT	
Eval	CIFAR-10	ImageNet	CIFAR-10	ImageNet
Inc in %	84.29	81.36	72.49	91.06

Our results indicate that higher coverage of perturbation is more likely to increase the effectiveness of the attack. Constraint-based technique is less effective than ILFO for both the datasets. Coverage of perturbation is one of the reasons for this. Constraint-based technique changes only 3 - 4 % of the total number of image pixels. For the CIFAR-10 dataset, the total number of pixels changed by Constraint-based technique is lower than the total pixels changed for the ImageNet dataset. This affects Constraint-based technique's effectiveness against the CIFAR-10 images. In another experiment, we have tried to change pixels of a small patch (5×5) on the CIFAR-10 images. The increase in computation does not increase beyond 5 - 10% irrespective of the positions of the patches.

Based on performance, ILFO is more effective on the CIFAR-10 dataset than the ImageNet dataset. The reason for this effect is that the scope of increasing computations is higher for the CIFAR-10 dataset than the ImageNet dataset. Another noticeable point is that the range of change in FLOPs after the attack for the CIFAR-10 images is larger than the range for the ImageNet images. The initialization of perturbation can be one of the reasons behind this. We use randomized initialization of perturbation, and for few inputs, the optimization can get stuck at a local minima. However, we can refer to the results that there is fewer number of inputs (represented as outliers) for which the increment of computation is low.

5.3.2 Quality

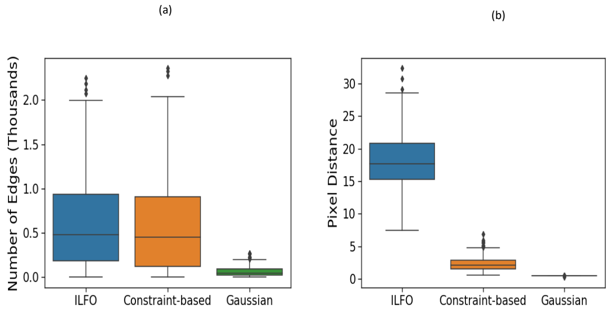


Figure 8. (a) Difference in number of edges detected in an perturbed image and original images after different attacks. (b) Difference in Euclidean distances between generated and perturbed images after different attacks.

We have measured the quality of an image by the syntactic and semantic values of an image. While implementing

Table 5. Time consumption of the attacks.

Eval	CIFAR-10		ImageNet	
	ILFO	Constraint-based	ILFO	Constraint-based
Time(sec)	229	278	800	560

ILFO, we apply more weightage on increasing computation than minimizing perturbation by choosing a high constant value (c). Because of this reason, the difference in quality between original and ILFO generated images can be higher than the Constraint-based technique generated image. To measure quality, we select 100 generated images by ILFO and Constraint-based technique. These images are generated by attacking SkipNet model, and the images are taken from the ImageNet dataset.

We have followed two types of measurement: syntactic difference and semantic difference. Euclidean distance of pixels between two images explains the syntactic difference between images. Canny Edge Detection (CED) and Fast Fourier Transform (FFT) have been used to measure the semantic difference of images. CED and FFT are traditional computer-vision algorithms that have been used by Jang *et al.* [17] to find out the effect of adversarial attacks on the features of the image. Canny Edge Detection (CED) is used to detect the change in the number of edges in the image after the attack. Fast Fourier Transform is used to convert signals from spatial or temporal domain to the frequency domain and to know change in the detailed feature after the attack (Details in Section B in Appendix).

Figure 8 and Figure 9 (in Appendix) show the comparison between quality of the original image and attack-generated images. Our results show that, in spite of having syntactic differences in generated images, the effects of both attacks on semantic values of the image are similar. In terms of Euclidean distance between image pixels of the original and generated image, the average change in pixel value for ILFO is highest (Figure 8 (b)). However, the difference in detected edges between the original and the attacked images (Figure 8 (a)) is similar for ILFO and Constraint-based technique (average difference of 616 and 582 respectively). Euclidean distance between generated FFTs also shows the difference in features in the images (Figure 9 in Appendix). The average distance measured with generated FFTs for ILFO is higher by a magnitude of 2 than the Constraint-based technique. All these results conclude that, in terms of semantic difference of the generated and original images, both attacks perform similarly.

5.3.3 Efficiency

We measure efficiency of an attack by measuring time consumption, and power consumption per second of the attack. We select ten CIFAR-10 and ImageNet images from the selected images during efficiency measurement experiments and attacked using ILFO and Constraint-based technique for SkipNet model. We use powerstat tool to measure CPU power consumption during both attacks, ensuring no other process uses CPU during the experiment. The number of iterations used for each technique is mentioned in Table 2. We have noticed that the attacks on the first image in the dataset are more time consuming than the attacks performed on later images. Because of this reason, we show the power consumption of both the attacks on the third image. Table 5 and Figure 10 (In Appendix) show the efficiency of two techniques for one image. Results in Figure 10 show that, with a similar amount of time and power consumption, ILFO performs larger number of iterations (700 more for CIFAR-10 and 290 more for ImageNet). Discussed results suggest that ILFO is more efficient than Constraint-based technique.

6. Conclusion

This paper presents ILFO [3], the first work investigating the robustness of neural networks in terms of energy consumption. ILFO opens intriguing new problems. First, formulation about the relation between input and its corresponding energy consumption can be further investigated given ILFO uses only the intermediate output as the proxy to formulate such relation. Second, ILFO can be used to develop universal defense for adversarial attacks. Instead of increasing the energy consumption, we can use ILFO to optimize images to the format that the energy consumption is minimum. Last, locating features affecting the energy consumption remains to be an open problem. One plausible solution is to divide the input data space into different regions and use genetic algorithm to search the most impactful features.

[3] https://sites.google.com/view/ilfo/home

References

[1] A. Bojchevski and S. Günnemann, “Adversarial Attacks on Node Embeddings via Graph Poisoning,” in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 695–704.

[2] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, “Adaptive Neural Networks for Efficient Inference,” in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 527–536.

[3] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, “Hidden Voice Commands,” in *Proceedings of the USENIX Security Symposium*, 2016, pp. 513–530.

[4] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, “Adversarial Classification,” in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 99–108.

[5] D.Lowd and C.Meek, “Good Word Attacks on Statistical Spam Filters,” in *Proceedings of the Second Conference on Email and Anti-Spam*, 2005.

[6] X. Dong, J. Huang, Y. Yang, and S. Yan, “More is Less: A More Complicated Network with Less Inference Complexity,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5840–5848.

[7] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, “Spatially Adaptive Computation Time for Residual Networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1039–1048.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples,” *arXiv preprint arXiv:1412.6572*, 2014.

[9] A. Graves, “Adaptive Computation Time for Recurrent Neural Networks,” *arXiv preprint arXiv:1603.08983*, 2016.

[10] J. Guan, Y. Liu, Q. Liu, and J. Peng, “Energy-efficient Amortized Inference with Cascaded Deep Classifiers,” *arXiv preprint arXiv:1710.03368*, 2017.

[11] S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep neural Networks with Pruning, Trained Quantization and Huffman Coding,” *arXiv preprint arXiv:1510.00149*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] Y. He, X. Zhang, and J. Sun, “Channel Pruning for Accelerating Very Deep Neural Networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[14] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[17] U. Jang, X. Wu, and S. Jha, “Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning,” in *Proceedings of the Annual Computer Security Applications Conference*, 2017, pp. 262–277.

[18] R. Jia and P. Liang, “Adversarial Examples for Evaluating Reading Comprehension Systems,” *arXiv preprint arXiv:1707.07328*, 2017.

[19] D. Karmon, D. Zoran, and Y. Goldberg, “LaVAN: Localized and Visible Adversarial Noise,” *arXiv preprint arXiv:1801.02608*, 2018.

[20] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning Filters for Efficient Convnets,” *arXiv preprint arXiv:1608.08710*, 2016.

[21] D. Lowd and C. Meek, “Adversarial Learning,” in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2005, pp. 641–647.

[22] F. Nan and V. Saligrama, “Adaptive Classification for Prediction Under a Budget,” in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 4727–4737.

[23] OpenAI, “AI and Compute,” https://openai.com/blog/ai-and-compute/.

[24] H. Qi, E. R. Sparks, and A. Talwalkar, “Paleo: A performance model for deep neural networks,” in *Proceedings of the International Conference on Learning Representations*, 2017.

[25] E. Strubell, A. Ganesh, and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," *arXiv preprint arXiv:1906.02243*, 2019.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[27] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks," *arXiv preprint arXiv:1312.6199*, 2013.

[28] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast Inference via Early Exiting from Deep Neural Networks," in *Proceedings of the International Conference on Pattern Recognition*, 2016, pp. 2464–2469.

[29] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning Dynamic Routing in Convolutional Networks," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 409–424.

[30] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[31] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial Attacks on Neural Networks for Graph Data," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 2847–2856.

Appendices

A. Constraint-based attack

Constraint-based technique is designed based on equation (3). This technique tries to minimize the distortion while constraints are responsible for increasing FLOps. Constraint-based technique can be time or resource consuming because it is an optimization technique with non-linear constraints. Because of that reason, we use a smaller input size. We send perturbation values as the input of the optimization technique. We only perturb selected pixels to keep the input size low. Constraint-based technique only perturbs pixels in the border of the image. Changing only border pixels makes sure that the main object of the image does not get perturbed. We divide the technique into four steps.

Initializing Output. This step represents initializing the output of the optimization function *i.e.,* resultant perturbation. The optimization with non-linear constraints may find itself in a local minima without proper Initialization. Initialization is done by randomly generating 100 values for distortion and selecting the value which generates the highest FLOPs increase and has the lowest absolute value.

Creating the constraints. In the equation (3), the constraint is designed to keep all α values $True$. But with our observations from results, we notice that all the blocks can not be made active simultaneously for every image Specially for a few images, the behaviours of two blocks are totally opposite, *i.e.,* making one block active makes other inactive. Because of this reason, instead of making all the blocks active, we have tried to make a certain number of blocks active. If N_G represents the number of active blocks in the model, then the constraint will be $N_G >= G_{Max}$ where G_{Max} is the predefined desired number of active blocks.

Creating Minimization Function. Minimization function has been designed to minimize the difference between the original and attacked image. For Constraint-based attack, we minimize the difference between pixel values of the original and distorted image.

Calling Optimization Function. The optimization function for Constraint-based perturbation attack can be represented as,

$$minimize(||x - x'||) \qquad (6)$$

such that,

$$(N(x') > G_{Max})$$

Where, N is a method that returns number of active gates, G_{Max} is desired active number of gates, x' is the attacked image with changed borderline pixel values, x is image input. We have used Sequential Least Squares Programming (SLSQP) as the optimization algorithm.

B. Quality Measurement Algorithms

Canny Edge Detection (CED) is a method to detect edges in an image. Pixels whose value is significantly different from their neighbors are called edges. We use edge detection as a metric to measure quality by analyzing the change in the number of edges in the image after the attack. Canny edge detection follows certain steps to detect edges. After reducing the noise from the image, CED calculates intensity gradients by convolving the image with predefined Sobel filters. We get the information about changes in values over pixels. Next, edge thinning is performed by suppressing the gradient, and finally, strong edges (higher gradient value) are selected to represent the edges of the image.

Fourier Transform is used to convert signals from spatial or temporal domain to the frequency domain. Fast Fourier Transform (FFT) is an algorithm to perform Fourier Transform with lower time complexity. Converted spatial frequency domain can be useful in giving useful information regarding image features. The rough shape structure of the image generates low spatial frequency after FFT. High spatial frequency corresponds to detailed features [17].

C. Euclidean distances between FFTs

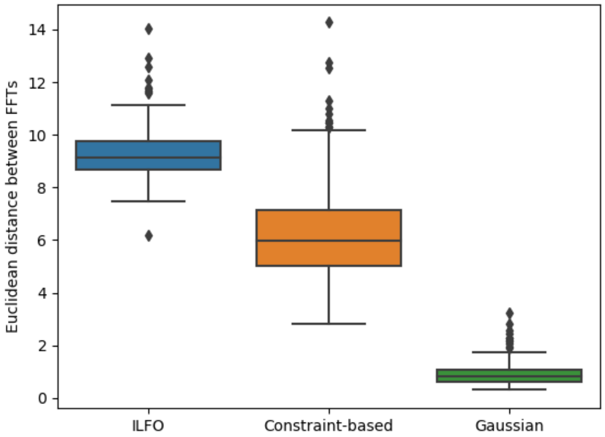
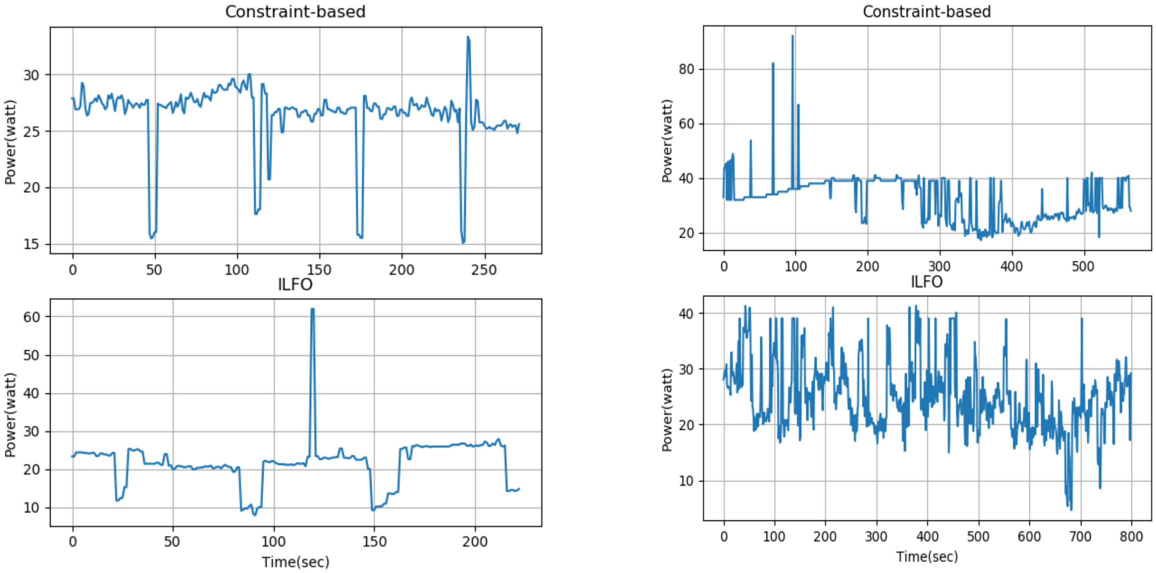


Figure 9. Euclidean distances between FFTs of generated and perturbed images

D. CPU power consumption of the techniques

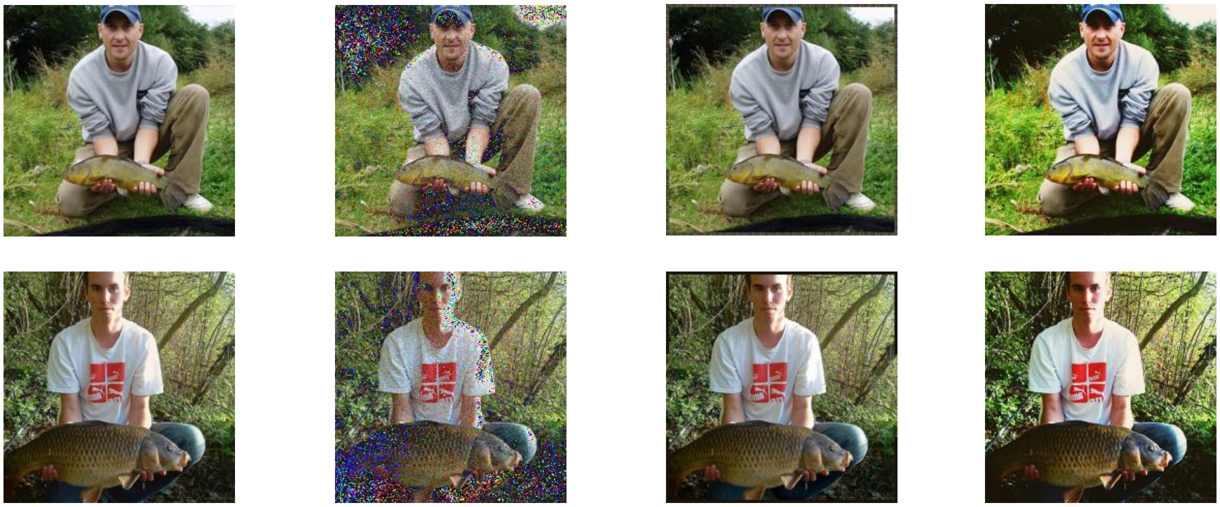
The results for CPU power consumption of different attack techniques is shown in Figure 10.



CIFAR-10

ImageNet

Figure 10. CPU power consumption of the attack techniques



Original

Gaussian Noise

Constraint-Based

ILFO

Figure 11. Effect of different attacks on ImageNet dataset images

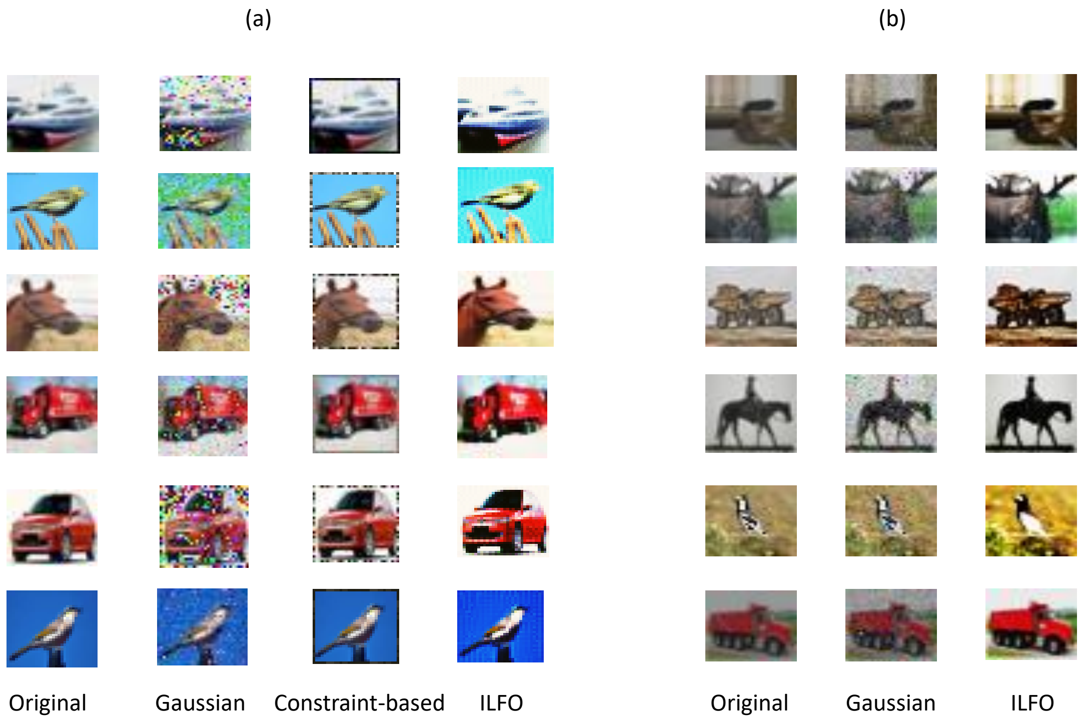


Figure 12. Effect of different attacks on CIFAR-10 dataset images.

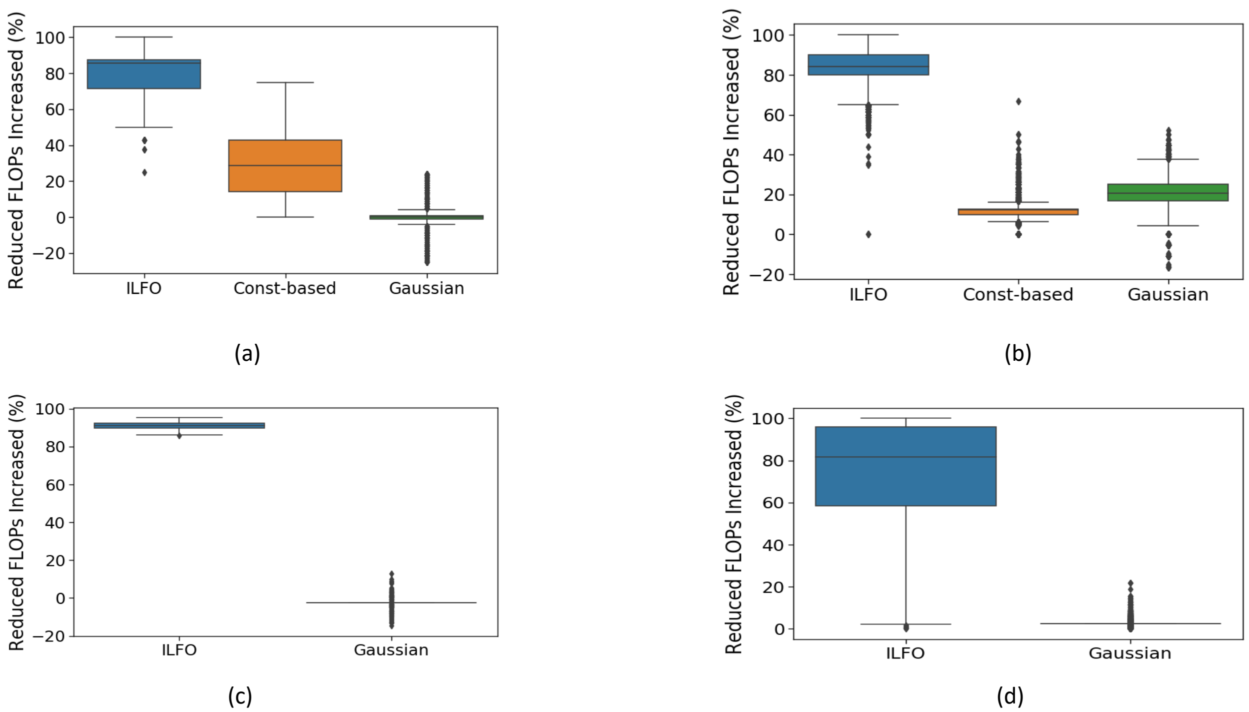


Figure 13. Percentage of reduced FLOPs increased by ILFO. (a) Results after attacking SkipNet using ImageNet images. (b) Results after attacking SkipNet using CIFAR-10 images. (c) Results after attacking SACT using ImageNet images. (d) Results after attacking SACT using CIFAR-10 images.