Dynamic Neural Network is All You Need: Understanding the Robustness of Dynamic Mechanisms in Neural Networks

Mirazul Haque, Wei Yang
University of Texas at Dallas, USA
mirazul.haque@utdallas.edu

Abstract

Deep Neural Networks (DNNs) have been used to solve different day-to-day problems. Recently, DNNs have been deployed in real-time systems, and lowering the energy consumption and response time has become the need of the hour. To address this scenario, researchers have proposed incorporating dynamic mechanism to static DNNs (SDNN) to create Dynamic Neural Networks (DyNNs) performing dynamic amounts of computation based on the input complexity. Although incorporating dynamic mechanism into SDNNs would be preferable in real-time systems, it also becomes important to evaluate how the introduction of dynamic mechanism impacts the robustness of the models. However, there has not been a significant number of works focusing on the robustness trade-off between SDNNs and DyNNs. To address this issue, we propose to investigate the robustness of dynamic mechanism in DyNNs and how dynamic mechanism design impacts the robustness of DyNNs. For that purpose, we evaluate three research questions. These evaluations are performed on three models and two datasets. Through the studies, we find that attack transferability from DyNNs to SDNNs is higher than attack transferability from SDNNs to DyNNs. Also, we find that DyNNs can be used to generate adversarial samples more efficiently than SDNNs. Then, through research studies, we provide insight into the design choices that can increase robustness of DyNNs against the attack generated using static model. Finally, we propose a novel attack to understand the additional attack surface introduced by the dynamic mechanism and provide design choices to improve robustness against the attack.

1. Introduction

Deep Neural Networks (DNNs) are used in multiple applications such as computer vision and natural language processing. After the rapid growth of IoT and embedded devices, many real-time systems use DNNs in their applications. As the real-time systems require faster response time and low energy consumption, researchers have proposed to incorporate energy-saving dynamic mechanism [34, 24, 36] to popular static DNN (SDNN) models like ResNet [20], VGG [31], MobileNet [22] etc. Early-exit is one of the dynamic mechanism techniques where multiple exits are included in SDNNs (creating multiple sub-networks), and SDNNs can terminate the operation early if a certain sub-network is confident about the prediction. These types of DNNs are named as early-exit Dynamic Neural Networks (DyNNs). Although the transition from SDNNs to DyNNs is preferred in real time systems because of increased efficiency, whether the use of dynamic mechanism will impact the robustness of the systems is still unknown. Studying the impact of the dynamic mechanisms on the robustness is important for developers or users to understand the trade-offs between DyNN and SDNN.

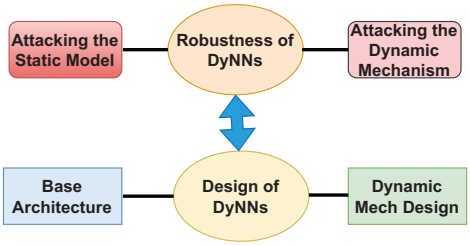


Figure 1: The overview of the objective of this work.

In this work, we propose to investigate how robustness is impacted through the addition of dynamic mechanism to the static model and how different dynamic mechanism designs would impact the robustness in dynamic mechanism (Figure 1). We propose to investigate two aforementioned topics through three research questions. These three RQs are based on: *Robustness of Dynamic Mechanism, Robust Design for the Static Model Attack, and Robust Design for the Dynamic Mechanism Attack.*

Robustness of Dynamic Mechanism. We focus on the investigation of robustness of the dynamic mechanism based on two aspects: transferability and the efficiency robustness.

First, we investigate the adversarial attack transferability between SDNNs and DyNNs to evaluate the robustness of the dynamic mechanism in black-box scenarios. In the black-box scenarios, adversaries normally assume the target models are always static. However, the target models can be dynamic also. Hence, it is important to find out if a surrogate SDNN model is used to attack a target DyNN model or vice-versa, then, to which extent the adversary can be successful.

To address this issue, in this paper, we first conduct a comparative study on the adversarial attack transferability between SDNNs and DyNNs (Section 3). Our study results suggest that adversarial transferability from DyNNs to SDNNs is better and surprisingly, using DyNNs as surrogate models for attack seems to be a more efficient and more effective way to generate adversarial samples. The adversaries are able to generate more adversarial samples in the same amount of time compared to using SDNNs as the surrogate model, and the generated adversarial samples often can also attack SDNNs.

The next robustness study is to understand how robust the efficiency of the dynamic mechanism is against the adversarial samples generated on SDNNs. This study tries to evaluate whether the original purpose of DyNNs (*i.e.,* saving inference time) will be impacted by the adversarial samples (Section 3.2) generated through SDNNs. Our study results suggest that the adversarial samples generated by existing white-box attacks and black-box attacks do not increase the inference time significantly, making the dynamic mechanisms efficiently robust against attacks on SDNNs.

Robust Design for the Static Model Attack Here, we perform a detailed analysis of which design choices in the dynamic mechanisms or DyNN architectures (specifically the position of early exits) may impact the robustness of DyNNs (Section 4). We consider two attack scenarios in this study: first, the output layer label of an SDNN is modified by a white-box adversarial example, and we study the impact of the example on corresponding DyNN's early-exit layers; second, in a black-box scenario, the output of SDNN is modified by a sample, and the sample is fed to separate model's DyNN. We have made multiple findings based on the empirical results, for example, putting the first exit earlier in the model architecture can help to improve the robustness of DyNNs.

Robust Design for the Dynamic Mechanism Attack. Last but not least, we evaluate the design choices of dynamic mechanism against a novel attack on dynamic mechanism. First, we design an adversarial attack approach to understand the extra attack surface introduced by the dynamic mechanisms in neural network (Section 5). In this attack, the synthesized adversarial examples will not change the prediction of the final output layer's label, but will change the prediction of all the early exits. Based on the evaluation results, we find that the dynamic mechanism is more vulnerable in scenarios where dependency among DyNN layers is lesser and when the exits are sparse w.r.t the layers.

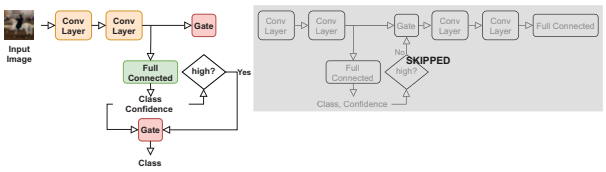


Figure 2: Working mechanism of Early-exit DyNN

Contribution. Our work provides insights about the robustness of the dynamic mechanism in DyNNs and also evaluates different design choices w.r.t robustness. The findings of this work would motivate developers/researchers to increase the usage of DyNNs and find more robust dynamic mechanisms.

2. Related Works and Background

Dynamic Neural Networks. The main objective of DyNNs is to decrease the energy consumption of the model inference for inputs that can be classified with fewer computational resources. DyNNs can be classified into *Conditional-skipping DyNNs* and *Early-exit DyNNs*. Early-exit DyNNs use multiple exits (sub-networks) within a single model and because of the model's working mechanism, the model is more suited for resource constrained devices. If, at any exit, the confidence score of the predicted label exceeds user defined threshold, inference is stopped. The resource-constrained devices usually deploy a lightweight sub-network of early exit network locally and resort to a server for further computations if needed [33] . [14], [12], [32], [24] have proposed Early-termination DyNNs. Specifically [24] and [38] propose early exit networks based on popular SDNNs. [38] also show that white-box robustness of the DyNNs is better than SDNNs. In addition to that multiple works [33, 29] provide practical usability of DyNNs.

Figure 2 shows the working mechanism of Early-exit DyNNs. For example, an Early-exit DyNN has N parts and each part has an exit. x is the input, f^i_{out} represents prediction after the i^{th} part (generated by specific computation unit), f_{out} represents prediction of the Neural Network, C_i represents confidence score after i^{th} part, Hid^{In}_i represents input of i^{th} part, Hid^{Out}_i represents output of i^{th} part, and τ_i is the predefined threshold to exit the network after i^{th} part. The working mechanism of the Early-exit DyNN can be represented as, $f_{out}(x) = f^i_{out}(x)$, if $C_i(x) \geq \tau_i$.

Adversarial Attacks. Adversarial Examples are the input that can change the prediction of the DNN model when those are fed to the model. [13] propose Fast Gradient Sign Method (FGSM) that uses single-step first order entropy loss

to generate adversarial inputs. This attack is modified by [27] to add initial noise to the benign sample. This attack is referred as projected gradient descent (PGD). Other than that, [10, 2, 9, **?**] have proposed white-box attack methods, while [26, 1, 23, 37] have proposed black-box attack methods. Recently, attacks [16, 19, 17, 18, 15, 4, 7, 6, 3] have been proposed to decrease the efficiency of the dynamic neural networks.

3. How robust are DyNNs against adversarial examples?

In this research question, we investigate the robustness of dynamic mechanism in DyNNs w.r.t adversarial examples generated on SDNNs. We further divide this question into two sub-parts: (i) attack transferability between SDNNs and DyNNs and (ii) efficiency robustness of DyNNs against adversarial examples generated on SDNNs. Through these RQs, we get an insight about how the existing adversarial attacks on SDNNs would impact the accuracy and efficiency of DyNNs. Also, we find out how adversarial examples generated on DyNNs impact the SDNNs w.r.t accuracy. As the SDNN models are static, adversarial examples would not have any impact on the efficiency of SDNNs.

3.1. Is adversarial example transferability from DyNN to SDNN lower than adversarial example transferability from SDNN to DyNN?

In this research question, we investigate the "transferability" of adversarial inputs generated based on SDNN and DyNN, respectively, *i.e.,* whether adversarial examples generated based on SDNNs are adversarial to DyNNs and vice versa. Transferability is an important metric for evaluating the feasibility of black-box attack. To evaluate the transferability, one of the popular way [28, 26] is creating a similar model (*i.e.,* surrogate model) as the target model. In a black-box attack, normally, adversaries assume the underlying model to be SDNN, so for a deployed DyNN, the adversaries may likely use an SDNN as surrogate model. Hence, this research question (RQ) is important to evaluate the robustness of DyNNs.

3.1.1 Experimental Setup.

Dataset and Models. We use CIFAR-10 and CIFAR-100 [25] datasets for evaluation. For SDNNs, we use VGG-16 [31], ResNet56 [20], and MobileNet [22] model. As DyNNs, we use the early exit version of these models [24]. In all other RQs, we keep the models and dataset setup same.

Black-box Attack. For the attack scenario, we use surrogate model [28, 26] based black-box attack scenario. Here, we feed a set of inputs to the target model and collect the output labels. These inputs are generated using 50% of the held-out validation data, and naturally corrupted versions of those validation data. As the number of partial held-out validation data is not significant, adding corrupted inputs would help to increase training data size for the surrogate model. For natural corruption [21], we use gaussian noise and brightness. For each type of corruption, we have five intensity levels. For example, if the number of held out data is 5000, for each corruption, we generate 25000 additional data. Once the input-output pairs are collected, a surrogate model is trained based on those pairs. For a target model, we use both SDNNs and DyNNs. If the target model is SDNN, then an DyNN is trained as surrogate model and vice-versa.

To make the surrogate-target pairs, we use different types of DNN architectures. For example, if the the target model is DyNN VGG, we choose ResNet56 SDNN as the surrogate model. This assumption is valid because the attacker doesn't have information about the target model architecture, hence the possibility of choosing the same architecture as the surrogate model is less. We define two terms to represent two different types of transferability based on different types of surrogate model and target model: **D2S** transferability and **S2D** transferability. D2S transferability evaluates DyNN to SDNN attack transferability, where S2D transferability evaluates SDNN to DyNN attack transferability. We have chosen following pairs to evaluate S2D transferability: (SDNN ResNet56 (surrogate), DyNN VGG (target)), (SDNN MobileNet (surrogate), DyNN MobileNet (target)), (SDNN MobileNet (surrogate),DyNN ResNet56 (target)). Similarly, for D2S transferability, earlier mentioned surrogate models become the target model and earlier mentioned target models become the surrogate model.

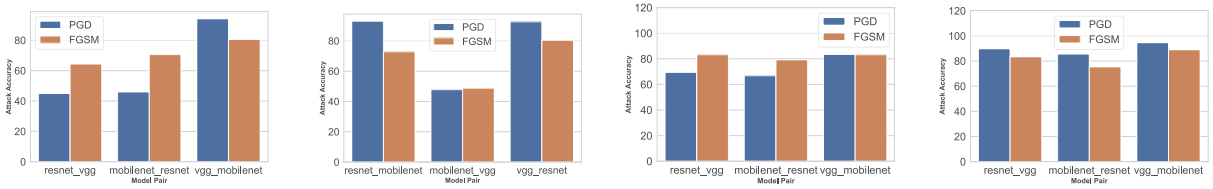
Algorithms. We use FGSM [13] and PGD [27] algorithms to attack the surrogate models.

Metric. We measure percentage of adversarial examples that can mis-classify the output w.r.t number of generated adversarial examples as the attack success rate.

3.1.2 Evaluation Results

Figure 3 shows the effectiveness of black-box attacks on DyNNs and SDNNs. On average, it can be noticed that for target SDNN and surrogate DyNN, the attack success rate is higher than the success rate of target AdNN and surrogate SDNN. One of the reasons for this behavior is the lower variance of the DyNNs. DyNNs use lower number of parameters, hence the feature space for adversarial samples of DyNNs is smaller than the feature space for adversarial samples of SDNNs [30]. Also, we find that for the target DyNN, FGSM attack performs better than PGD attack. If only dataset-specific results are considered, then for CIFAR-100 the attack success rate is higher than for CIFAR-10.

Also, as the DyNN-generated adversarial inputs can attack SDNNs, then it can be time efficient to create adversarial inputs using DyNN. Through Figure 8 (Appendix), we can



(a) Target DyNN (CIFAR-10) (b) Target SDNN (CIFAR-10) (c) Target DyNN (CIFAR-100) (d) Target SDNN (CIFAR-100)

Figure 3: Transferable Attack Success Rate for CIFAR-10 and CIFAR-100 data

find the probability density plots of different exit numbers of DyNN that have been used to generate adversarial examples. Lower exit number suggests that lesser number of computations has been used to generate adversarial examples. It can be noticed that for PGD attack, more than 70% of adversarial examples are generated from exit 0, 1 and 2 (first - third exit). Although for FGSM attack, in a few scenarios (CIFAR-10 VGG, CIFAR-100 MobileNet, and CIFAR-100 ResNet), more than 50% of adversarial examples are generated through later exits (higher computation required).

We also perform some additional experiments to strengthen our claim. In appendix, Figure 13 shows the experimental results on MI-FGSM attack, which is also a highly transferable attack. Also Figure 17 shows the results on Tiny-ImageNet dataset, which has a higher input size than CIFAR datasets. These two results show that our claim also holds for addditional attacks and datasets. Also Figure 14,15 and 16 in appendix shows adversarial examples generated through SDNNs and DyNNs. We find that both type of adversarial examples are similar w.r.t quality.

Finding 1: *The **D2S** transferability is higher than the **S2D** transferability.*
Finding 2: *Using DyNNs as surrogate models is more efficient and more effective way to generate adversarial examples than using SDNNs.*

3.2. Does adversarial examples impact efficiency robustness in DyNNs?

In this section, we investigate if the dynamic mechanism in DyNNs is robust w.r.t efficiency against the adversarial examples generated on static mdoel. Through the evaluation, we ensure whether the original purpose of including dynamic mechanism in DyNNs (*i.e.,* saving inference time) will be impacted by the adversarial samples. Specifically, we study whether the adversarial inputs exit earlier or later in a DyNN compared to the original inputs. The main objective of this investigation is to find out whether the adversarial samples generated on SDNN can have an impact on the amount of computation of the DyNN. For this purpose, we conduct both white-box and black-box attacks to find out the impact of adversarial samples on the amount of computation.

Here, the white-box attack scenario can also be considered a practical scenario. There have been studies [5, 35] that focus on reverse engineering of SDNN models from binary code of on-device models, but no techniques have been proposed to reverse engineer the dynamic mechanisms in the models. Hence, adversaries are more likely to get SDNN models instead of their dynamic counterparts. So it is important to find out how adversarial samples affect the efficiency of the DyNN for both white-box and black-box scenarios.

3.2.1 Experimental Setup.

Attack. We use PGD and FGSM for both white-box and black-box attacks. For black-box setup, we use the same setup as previous RQ. For black-box scenario, we use DyNNs as target model and SDNNs as surrogate model. In a white-box setting, we attack the SDNN and evaluate on the performance of corresponding DyNN.

Metric. We use the difference between the exit number selected by adversarial input and the exit number selected by benign input. If the difference is positive, then the latency of the adversarial sample is increased w.r.t benign input.

3.2.2 Evaluation Results.

Figure 4 and Figure 11 (in Appendix) show the impact of adversarial examples generated on SDNN on changing in exit number in DyNN in a white-box setting. It can be observed that for the majority of the scenarios, accuracy-based adversarial samples do not increase the computation significantly in the DyNN. On average, FGSM-generated examples increase more computation in DyNNs than PGD-generated examples. For CIFAR-10 data, for MobileNet and VGG-16 DyNN models, 25%-37% of the FGSM generated examples could increase the number of exits by more than one. Also, it can be noted that adversarial samples generated on CIFAR-100 data is more likely to increase computation than adversarial samples generated on CIFAR-10. Especially, more than 45% of the FGSM samples generated on CIFAR-100 data can increase the number of exits by more than one.

Figure 5 and Figure 12 (in Appendix) show the impact of adversarial examples generated on SDNN on change in exits in DyNN in a black-box setting. It can be noted that, for CIFAR-10 dataset, black-box attack can generate more computation-increasing examples than in white-box attack. For ResNet and VGG model, more than 40% PGD attack generated examples can increase the number of exits by more than one. For all three models, 35% of the FGSM attack generated examples can increase the number of exits by more than one. FGSM attack generates more inputs that induces low confidence in early-exit layers than PGD attack. For CIFAR-100 dataset, the increase of computation caused by adversarial attacks is higher than that of CIFAR-10. As CIFAR-100 data uses larger model, the robustness of the model is reduced. For CIFAR-100 dataset, more than 40% examples generated through both the attack can increase the number of exits by more than one. However, increasing the number of exits by two or three exits does not decrease the efficiency in DyNNs significantly.

Finding 3: *Accuracy-based adversarial samples do not decrease the efficiency significantly in the DyNN.*
Finding 4: *The adversarial examples, whose output confidences are significantly lower, can perform better in terms of decreasing the efficiency in DyNNs.*
Finding 5: *Adversarial examples generated on a larger model (w.r.t model parameters) is more likely to decrease efficiency in DyNNs.*

4. What design of DyNNs may impact the robustness against attack generated on SDNNs?

In this section, we evaluate which architecture design choices (position of early exits) in DyNNs may impact the robustness of early layers against the adversarial inputs generated on SDNNs. In this RQ, we consider DyNNs as multi-exit networks, where each exit will provide an output. For evaluation, we assess if we attack the final exit (output of the static model), from which early-exit layer the label modification begins. If the output label is not modified in any earlier layer, then for that type of model, the robustness is higher because the model can produce correct results at any layer. This RQ will provide us an insight into which type of design choice may improve the robustness of early layers.

4.1. Experimental Setup.

Attack Setup. We use same attack setup as previous research questions. However, while we attack the DyNN, we do not consider one specific exit layer. Instead of that, we consider each exit layer.

Metric. We use the early exit from which the label modification starts. For example, there are N exits. First, Nth exit's label is modified through adversarial sample. If till Kth exit the original prediction was same, then we report $K+1$th exit in the experimentation.

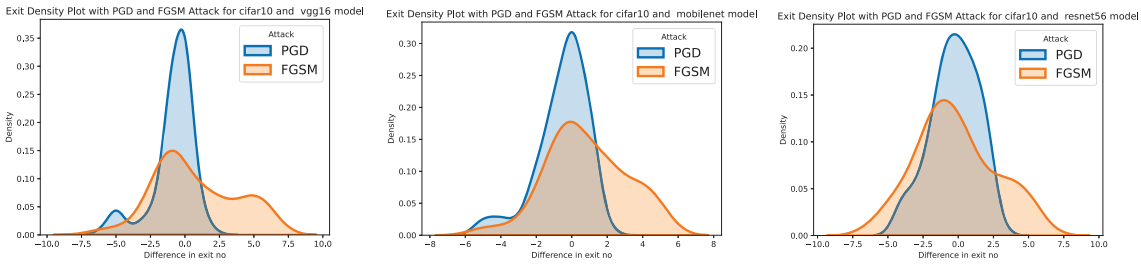
4.2. Evaluation Results.

Figure 6 and Figure 9 (in Appendix) show probability density plot on which exit the output label is changed using the white-box adversarial examples. It can be observed that for all the model-dataset pairs, for more than 77% of the examples, the label is modified in the first exit. For CIFAR-10 data, only for MobileNet and ResNet models, the label is changed after the first exit for more than 20% of the examples (using FGSM attack).

Figure 7 and Figure 10 (in Appendix) show probability density plot on which exit the output label is changed using the black-box adversarial examples. From results, we can see that the robustness of earlier exits is better against black-box attack than white-box attack. For CIFAR-10 data, more than 45% of the both attack generated samples could not misclassify the first exit for VGG-16 model. For CIFAR-100 data (larger model), robustness of early exit layers is worse than that of CIFAR-10 data (smaller model). For CIFAR-100 data, for both attacks and three models, less than 30% of the adversarial examples can not mis-classify the first exit.

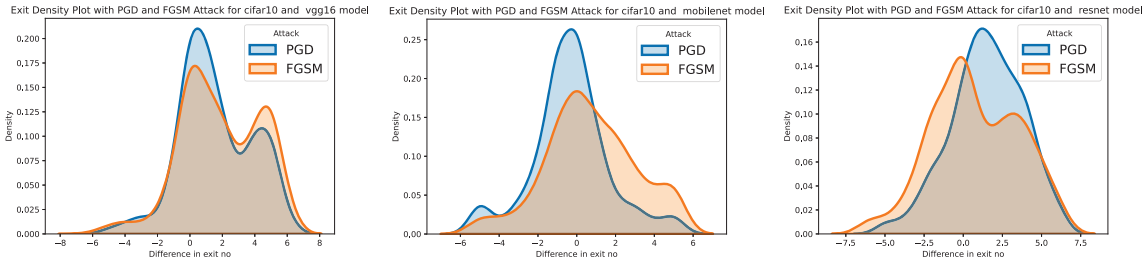
For black-box attack scenario, VGG-16 model's first layer on-average shows better robustness against black-box attack. In the DyNN, the first exit of the VGG model is placed only after second layer, while for others, more computations are performed before using the first exit. Hence, having the first exit in the early layers can increase the robustness of DyNN. Although lower number of parameters would be used to predict output, but with VGG we can notice that a significant number of inputs can be classified correctly through first exit.

Finding 6: *We find that having the first exit in the earlier layers can increase the robustness of early exits of the DyNNs.*
Finding 7: *Black-box attack success rate is lesser than white-box attack success rate against early-exit layers.*
Finding 8: *For black-box scenario, early-exit layers are more robust against adversarial examples generated on a smaller surrogate model than adversarial examples generated on a larger surrogate model.*



(a) CIFAR-10 VGG (b) CIFAR-10 MobileNet (c) CIFAR-10 ResNet

Figure 4: Density plots of change in exit numbers because of PGD and FGSM attack (For CIFAR-10 data). The x axis represents the change in exit number while y axis represents the density.



(a) CIFAR-10 VGG (b) CIFAR-10 MobileNet (c) CIFAR-10 ResNet

Figure 5: Density plots of change in exit numbers because of PGD and FGSM **black-box** attack (For CIFAR-10). The x axis represents the change in exit number while y axis represents the density.

5. What design of DyNNs may impact the robustness against attack on dynamic mechanism?

In this section, we evaluate if specific examples can be generated only to understand the additional attack surface introduced by the dynamic mechanism in DyNNs and which design choices would be helpful in DyNNs to improve the robustness against this specific attack. First, we aim to design an attack such that the synthesized adversarial examples will not change the prediction of the final SDNN label, but change the prediction of all the early exits. This threat model is practical because the attack evades the existing detection that relies on the final output of SDNN while the attacker creates a situation where all the early exit networks do not provide correct prediction, therefore decreasing the usability of DyNNs. However, this attack is also challenging to be performed successfully because final layer output is dependent on the earlier exit layers and it is challenging to impact all the early exits' prediction without modifying the final exit's prediction.

5.1. Problem Formulation

We propose a novel attack technique called *Early Attack* to evaluate the effectiveness of the early layers of DyNNs.

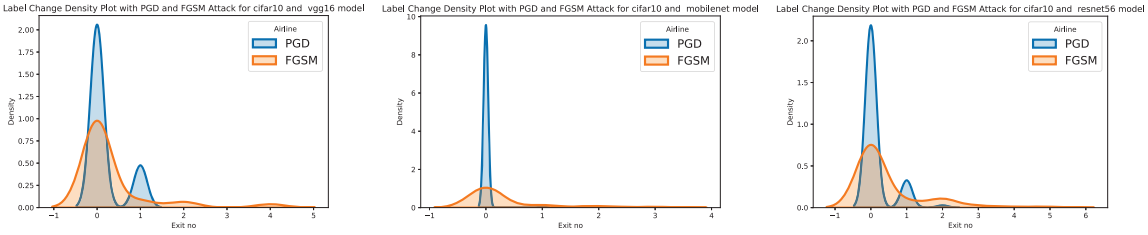
Algorithm 1: Input generation using Early Attack

Inputs : $x : Input\ Image$
Outputs : $x' : Perturbed\ Image$

begin
  Initialize(w)
  $T = number\_of\_iterations$
  $iter\_no = 0$
  while $iter\_no < T$ **do**
    $x' = \frac{\tanh(w)+1}{2}$
    $output = model(x')$
    if $success(output)$ **then**
      $return(x')$
    end
    $L = loss(x, w, c, \alpha)$
    $L_{new}, w = Optim(L, w)$
    $iter\_no++$
  end
  $x' = \frac{\tanh(w)+1}{2}$
end

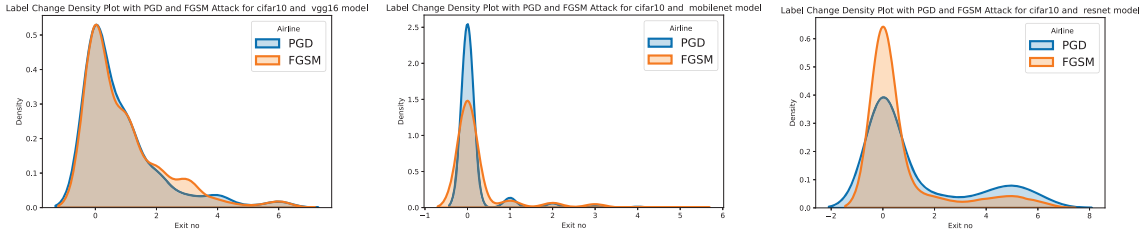
Let's assume f is an DyNN with N exits. Given an input x, the output softmax layer in an exit i can be defined as $y^i = f_i(x)$, where $i = 1, 2, 3...N$. For synthesizing adversarial examples, we have two main objectives. First, the initial prediction in the N^{th} exit (final layer) does not change. Let's assume, the initial prediction at final layer is p. Second, the prediction of all the other exits should be different than p.

Based on the aforementioned objectives, we can propose



(a) CIFAR-10 VGG (b) CIFAR-10 MobileNet (c) CIFAR-10 ResNet

Figure 6: Density plots representing during which exit number output label is changed because of PGD and FGSM attack (For CIFAR-10 data). The x axis represents the exit number while y axis represents the density.



(a) CIFAR-10 VGG (b) CIFAR-10 MobileNet (c) CIFAR-10 ResNet

Figure 7: Density plots representing during which exit number output label is changed because of PGD and FGSM **black-box** attack (For CIFAR-10 data). The x axis represents the exit number while y axis represents the density.

an iterative optimization procedure to optimise a loss function L. For each of the objectives, one loss function component is formulized. For the first objective, we propose loss function $L1 = (-1 * \sum_{j \neq p} max(y_p^N - y_j^N, 0))$. In $L1$, we maximize the difference between softmax value of label p and other label's softmax value, therefore the prediction won't get changed at the final exit. For the second objective, we propose loss function $L2 = (\sum_{i=1}^{N} y_p^i)$. The $L2$ ensures that for any other exit than the final exit, p's softmax value would be minimized. The final loss function $L = \alpha * L1 + L2$. Here the α is a user-defined variable that provides balance between two loss terms.

Finally, we need to ensure the added perturbation to generate adversarial examples are limited, hence, the final optimization function would be, $minimize(||\delta|| + c \cdot L)$, where, $(x + \delta) \in [0, 1]^n$. Here, δ represents the added perturbation and c is a user-defined variable to provide weightage on a specific component. c controls the magnitude of generated perturbation ($||\delta_i||$), where a large c makes the loss function more dependant on the L.

This constrained optimization problem in δ can be converted into a non-constrained optimization problem in w, where the relationship between δ and w is: $\delta = \frac{\tanh(w)+1}{2} - x$ The $tanh$ function would ensure that the generated adversarial input values stay between 0 and 1. The equivalent optimization problem in w is:

$$\underset{w}{\text{minimize}} \left|\left| \frac{\tanh(w) + 1}{2} - x \right|\right| + c \cdot L \qquad (1)$$

Algorithm 1 shows the optimization algorithm. The algorithm outputs the adversarial input x' given a benign image x as input. w is initialized to a random tensor that has equal shape as the input image. For each iteration, the loss function of the attack is computed (at line 11). Based on the back-propagated loss, the optimizer updates w with its next value. Once the iteration threshold (T) is reached or the attack is successful, the algorithm computes and returns the adversarial input x' (at Line 9 and 15).

5.2. Evaluation

5.2.1 Evaluation SetUp.

Baseline. We use PGD and FGSM attacks as baseline to modify the early exit label.

Metric. We use attack success rate as metric in the evaluation. If the adversarial input generated final layer output is same as the output generated by benign input and all the other exit layers output is different than the final output label, then we consider attack is successful for that particular adversarial input.

Hyperparameters. We use $\alpha = \{0.001, 0.01, 0.1, 1, 20, 40\}$ and $c = 50$ as hyperparameters.

5.2.2 Effectiveness

Table 1 shows the evaluation results of the attack success rate of Early Attack and baseline techniques. Except two model-dataset pair, Early Attack's success rate is higher than 80% in all other scenarios. PGD and FGSM attacks are unsuccessful because the loss function only considers early layer output and does not consider final layer output. Although FGSM's attack success rate is higher than PGD's success rate. Also, w.r.t best performing α values, $\alpha = 1$ performs well for 70% of the model-dataset pairs.

We also analyze why early attack fails for VGG16 and MobileNet models for CIFAR-10 data. First, for CIFAR-10 data, final fully connected layer has lower number of parameters than the final layer for CIFAR-100 data. Hence, the dependency between the final layer and earlier layers is higher for CIFAR-10. However, for CIFAR-10 data, attack success rate is higher for ResNet model. First we discuss the failed cases for VGG16 and MobileNet model, and then we discuss why the attack succeeded for ResNet model.

For VGG16 model, the failed examples can be divided into two types. For the first type, the final output label is changed with all the other early exit layers. In the second type, first few layers get the output label mis-classified, but along with the final layer, few previous layers also get the output correctly classified. For MobileNet, all the layers except the final and last to final layer get the output label mis-classified. For MobileNet, we also find that the final two exits are separated by only two layers, which is significantly lower than other model's separation layers between final two exits. Hence, the dependency between final two layers is higher.

For ResNet model, there are 56 layers divided into 27 blocks. The exits are sparsely divided between these blocks. For Mobilenet and VGG models, the exit distributions are less sparse. Hence, the dependency between exits is lower for ResNet and because of that reason, we could successfully attack ResNet model.

Table 1: **Attack accuracy percentage of Early Attack and the baseline techniques against different model and dataset, along with α value.**

Dataset	Model	Early Attack	best α val	PGD	FGSM
CIFAR-10	VGG	**35**	1	0	0
	MobileNet	**11**	0.1	0	0
	ResNet	**81**	1	0	0
CIFAR-100	VGG	**86**	20	0	1
	MobileNet	**97**	1	0	0
	ResNet	**96**	1	0	2

5.2.3 Transferability

In this section, we discuss about the transferability of the Early Attack examples. As Early Attack has two different components, instead of measuring attack success rate directly, we measure two parameters $T1$ and $T2$. $T1$ represents the percentage of inputs for which the final output remains same as the output generated by benign input. $T2$ represents from the examples selected from $T1$, how many early exit layers on average are mis-classified. Having both high $T1$ and $T2$ values ensures transferability.

Table 2: $T1$ **and** $T2$ **values for measuring transferability between three models.** ***TM*** **represents Target Model and** ***SM*** **represents Surrogate Model.** $T1$ **presents the percentage of inputs for which the final output remains same as the output generated by benign input.** $T2$ **represents from the examples selected from** $T1$**, how many early exit layers on average is mis-classified.**

		CIFAR-10			CIFAR-100		
Type	SM \ TM	VGG	MNet	RNet	VGG	MNet	RNet
$T1$	VGG	–	85%	73%	–	65%	39%
	MNet	86%	–	74%	51%	–	38%
	RNet	89%	82%	–	79%	75%	–
$T2$	VGG	–	0.73	1.65	–	1.58	2.69
	MNet	1.06	–	2.3	1.52	–	3.26
	RNet	0.95	0.85	–	1.32	1.28	–

We show the transferability results in Table 2. For CIFAR-10 data, the $T1$ values are high, but $T2$ values are significantly low except for MobileNet to ResNet transferability. For CIFAR-100 data, $T2$ values are higher than of CIFAR-10 data, however, $T1$ values are low. From the results, we can notice that the generated examples either can keep the final layer label the same or can change the early exit layer outputs. Our finding suggests that early attack transferability is limited.

> **Finding 9**: *With increasing number of exits in DyNNs, the dependency between multiple exits will increase. Hence, more exits in DyNNs can increase the robustness against the Early Attack.*
> **Finding 10**: *Early Attack transferability between DyNNs is not significant.*

6. Conclusion

In this work [1], we discuss the robustness of including dynamic mechanism in DNN through three research questions. We find out that DyNNs are more robust than SDNNs and also efficient to generate adversarial examples. We also propose DyNN design choices through final two RQs. Finally, we propose a novel attack to understand additional attack space in DyNNs.

[1] https://github.com/anonymous2015258/Early_Attack/tree/main

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[3] Simin Chen, Hanlin Chen, Mirazul Haque, Cong Liu, and Wei Yang. The dark side of dynamic routing neural networks: Towards efficiency backdoor injection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24585–24594, 2023.

[4] Simin Chen, Mirazul Haque, Cong Liu, and Wei Yang. Deepperform: An efficient approach for performance testing of resource-constrained neural networks. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13, 2022.

[5] Simin Chen, Hamed Khanpour, Cong Liu, and Wei Yang. Learning to reverse dnns from ai programs automatically. *arXiv preprint arXiv:2205.10364*, 2022.

[6] Simin Chen, Cong Liu, Mirazul Haque, Zihe Song, and Wei Yang. Nmtsloth: understanding and testing efficiency degradation of neural machine translation systems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1148–1160, 2022.

[7] Simin Chen, Zihe Song, Mirazul Haque, Cong Liu, and Wei Yang. Nicgslowdown: Evaluating the efficiency robustness of neural image caption generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15365–15374, 2022.

[8] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

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

[10] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[11] Fernando A Fardo, Victor H Conforto, Francisco C de Oliveira, and Paulo S Rodrigues. A formal evaluation of psnr as quality measurement parameter for image segmentation algorithms. *arXiv preprint arXiv:1605.07116*, 2016.

[12] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially Adaptive Computation Time for Residual Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2017.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014.

[14] Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *arXiv preprint arXiv:1603.08983*, 2016.

[15] Mirazul Haque, Christof J Budnik, and Wei Yang. Corrgan: Input transformation technique against natural corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 194–197, 2022.

[16] Mirazul Haque, Anki Chauhan, Cong Liu, and Wei Yang. ILFO: Adversarial Attack on Adaptive Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14264–14273, 2020.

[17] Mirazul Haque, Simin Chen, Wasif Arman Haque, Cong Liu, and Wei Yang. Nodeattack: Adversarial attack on the energy consumption of neural odes. 2021.

[18] Mirazul Haque, Rutvij Shah, Simin Chen, Berrak Şişman, Cong Liu, and Wei Yang. Slothspeech: Denial-of-service attack against speech recognition models. *arXiv preprint arXiv:2306.00794*, 2023.

[19] Mirazul Haque, Yaswanth Yadlapalli, Wei Yang, and Cong Liu. Ereba: Black-box energy testing of adaptive neural networks. *arXiv preprint arXiv:2202.06084*, 2022.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[21] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

[23] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.

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

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

[26] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

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

[28] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[29] Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966, 2020.

[30] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *arXiv preprint arXiv:1808.05665*, 2018.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast Inference via Early Exiting from Deep Neural Networks. In *Proceedings of the International Conference on Pattern Recognition*, pages 2464–2469, 2016.

[33] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pages 328–339. IEEE, 2017.

[34] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning Dynamic Routing in Convolutional Networks. In *Proceedings of the European Conference on Computer Vision*, pages 409–424, 2018.

[35] Ruoyu Wu, Taegyu Kim, Dave (Jing) Tian, Antonio Bianchi, and Dongyan Xu. DnD: A Cross-Architecture deep neural network decompiler. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2135–2152, Boston, MA, Aug. 2022. USENIX Association.

[36] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[37] Guanqun Yang, Mirazul Haque, Qiaochu Song, Wei Yang, and Xueqing Liu. Testaug: A framework for augmenting capability-based nlp tests. *arXiv preprint arXiv:2210.08097*, 2022.

[38] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.