A TALE OF TWO MODELS

CONSTRUCTING EVASIVE ATTACKS ON EDGE MODELS

Wei Hao [1] **Aahil Awatramani** [2] **Jiayang Hu** [1] **Chengzhi Mao** [1] **Pin-Chun Chen** [1] **Eyal Cidon** [3 †] **Asaf Cidon** [1] **Junfeng Yang** [1]

ABSTRACT

Full-precision deep learning models are typically too large or costly to deploy on edge devices. To accommodate to the limited hardware resources, models are adapted to the edge using various edge-adaptation techniques, such as quantization and pruning. While such techniques may have a negligible impact on top-line accuracy, the adapted models exhibit subtle differences in output compared to the original model from which they are derived. In this paper, we introduce a new evasive attack, DIVA, that exploits these differences in edge adaptation, by adding adversarial noise to input data that maximizes the output difference between the original and adapted model. Such an attack is particularly dangerous, because the malicious input will trick the adapted model running on the edge, but will be virtually *undetectable* by the original model, which typically serves as the authoritative model version, used for validation, debugging and retraining. We compare DIVA to a state-of-the-art attack, PGD, and show that DIVA is only 1.7–3.6% worse on attacking the adapted model but 1.9–4.2× more likely not to be detected by the the original model under a whitebox and semi-blackbox setting, compared to PGD.

1 INTRODUCTION

Deep learning (DL) models are increasingly being deployed in large-scale applications on millions of edge devices, such as phones and cameras. Notable real-world edge deployments include language translation (Good, 2015), object detection (Ahmadyan & Hou, 2020), face recognition (fd-, 2017) and ad recommendations (Wu et al., 2019). The common lifecycle of edge-distributed DL models is to assemble a large dataset, and train the model on a powerful set of servers, using DL accelerators (e.g., GPUs, TPUs). After the model has been trained, it is typically pushed to a set of edge devices, where it will conduct inference (Guo et al., 2021; tfl, 2021; Wu et al., 2019). Edge devices are resource constrained compared to the servers used to train the model: they have weaker accelerators, less DRAM, limited power and energy constraints. Therefore, the model needs to be *adapted* to run on these devices.

To adapt models to resource-constrained edge devices, several techniques are commonly used, including quantization, distillation and pruning (Deng et al., 2020). Such techniques shrink the size and representation of the original model. For example, quantization converts the floating-point representation of the original model to a more compact form, such as int8 or int4, which reduces the computational complexity, the amount of memory, and the energy spent when conducting inference. While quantization generally reduces the accuracy of the model, on average, its effect is relatively modest; in our experiments the the adapted int8 version achieves at least 96% of the accuracy of the original model on average across three standard computer vision models. However, despite the small differences in top-line accuracy, the models are not identical, and may return slightly different results for a particular input.

Our key observation is that *subtle differences between the edge-adapted and the original models can be exploited by an attacker*. We propose DIVA (DIfferential eVasive Attack), an attack against computer vision models, which causes the edge-adapted model to mispredict, while remaining virtually undetectable to the original model. In large-scale production environments, ML operators run thousands of slightly-different adapted models on tens of thousands of device models (Wu et al., 2019). Such an attack would make it very difficult for the operator to detect and debug, because when validating the input against their authoritative original model they would not detect the input as malicious. Therefore, the operator may assume that no attack occurred until the attackers have subverted a significant portion of the edge devices. Even if the operator detected it, it would be expensive and time-consuming to debug the root cause and understand which edge models it affects.

[1]Columbia University [2]Cornell University [3]Meta †This research was done independently of Meta. Correspondence to: Wei Hao <wh2473@columbia.edu>.

Proceedings of the 5th MLSys Conference, Santa Clara, CA, USA, 2022.

DIVA constructs an attack image (indistinguishable to a human from the original image), with the goal of maximizing the prediction loss of the adapted model, while minimizing the prediction loss of the original model. DIVA uses simple iterative optimization methods, such as stochastic gradient descent, to efficiently construct the attack image. This allows DIVA to create an input that would cause the adapted model to mispredict, while causing minimal changes to the prediction of the original model. We initially design DIVA assuming a whitebox setting, where the attacker has access both to the original and the adapted model. We leverage differential testing (McKeeman, 1998; Pei et al., 2017), a powerful technique for detecting deviations of two implementations of the same functionality to generate adversarial samples. Compared to searching through a vast space of possible behaviors of one model, DIVA focuses on a much smaller space of deviations of two very similar models, and is therefore very effective at generating successful attacks. We show DIVA generalizes across two popular edge-adaptation techniques: quantization and pruning.

Since it may be much easier for the attacker to obtain access to the adapted model, which could be running on any edge device (e.g., phone, camera), rather than the original model, which might be stored in a more secure location (e.g., in the cloud), we also create a semi-blackbox attack for quantized models that requires only the adapted model. To this end, we reconstruct a full-precision surrogate model that shares the same architecture with the adapted model, by teaching the surrogate model to imitate the adapted model via knowledge distillation (Ba & Caruana, 2014; Polino et al., 2018). We then generate the adversarial attack on this surrogate model along with the adapted model and evaluate its efficacy on the original full-precision and the adapted models. Finally, we test DIVA in a blackbox setting where the hacker does not have access to the parameters of the adapted model as well and tries to reconstruct both a surrogate full-precision and a surrogate adapted model via knowledge distillation followed by adaptation.

Our evaluation shows that both whitebox and semi-blackbox DIVA significantly outperform a state-of-the-art attack, PGD (Madry et al., 2018), in causing the adapted model to mispredict while not affecting the original model. The whitebox DIVA attack is able to cause the adapted model to mispredict and original model to predict correctly 92.3–97% for quantization and 98–100% for pruning, across three architectures (ResNet, MobileNet, DenseNet) on ImageNet, while the semi-blackbox DIVA attack does so with a success rate of 71.1–96.2%. Even fully blackbox DIVA outperforms PGD, albeit with a lower success rate of 30.3–77.2%. Furthermore, despite the constraint not to affect the original model, DIVA is only up to 3.6% worse than PGD in misleading the edge models. Therefore, from the attacker's standpoint, there is very little cost in using DIVA compared to an attack that targets solely edge models.

Finally, to showcase a real-world attack scenario, we demonstrate how DIVA can be used to attack a face recognition model that would be running on a security camera or a phone. Similar to the ImageNet experiment, the whitebox DIVA attack achieves a 98% success rate here.

We make the following contributions:

1. We present a new vulnerability introduced by the widespread use of model adaptation, which causes slight variations between the models trained on the server and conducting inference on the edge.
2. DIVA is a new attack that targets adapted computer vision models running on edge devices, which is virtually-undetected by validation on the original model. We show that adding this evasive property to adverserial attacks comes at a low cost for the attacker.
3. DIVA is effective even in a semi-blackbox setting where the attacker can only access the adapted model. We design a novel attack where the attacker tries to reconstruct the original model using distillation learning, and is able to successfully attack the adapted model relatively undetected.
4. We show DIVA can be used in a realistic scenario of a face recognition model deployed on an edge device.

2 Background and Closely Related Work

This section presents a primer on model quantization, pruning and distillation (§2.1), surveys the latest adversarial attacks against DL models (§2.2) and defenses (§2.3), and introduces differential testing and its application to DL (§2.4), which uses techniques relevant to our work.

2.1 Quantization, Pruning and Distillation

Basic quantization. DL models are trained using a floating point representation with 32 or 64 bits (fp-, 2021; nvi, 2021). These floating point representations are typically supported by server-based accelerators (nvi, 2021; Rocki et al., 2020). When deployed to edge devices, models are often quantized in order to reduce their resource consumption. Quantization reduces the size of the model and its resource consumption when conducting inference, by representing the model using a smaller number of bits for each number (e.g., 8 bits instead of 32). This typically requires changing the representation from a floating point to an integer. While quantization reduces the compute, memory, and energy footprint, as well as the latency of inference (Cai et al., 2017; Han et al., 2016; Lin et al., 2016), it also reduces the model's accuracy.

Quantization-aware training. Quantization-aware training (QAT) (Jacob et al., 2018) is a technique that improves

the accuracy of the adapted model by introducing quantization noise *during the training of the model*. In QAT the base model training is still done in full precision. During the forward pass, weights and data are adapted to a lower precision (e.g., from float32 to int8) and increased back to the original precision, in the backpropagation the gradients are applied to the weights via straight-through estimators (Bengio et al., 2013). This forces the base model to learn weights that are more robust to the quantization error.

Pruning. Pruning is a technique that zeros out model weights that are insignificant and barely used during the training process to achieve sparsity. A sparse model has the advantages of easier compression, smaller model size and higher inference speed, resulting in a smaller and faster neural network. This technique is being used for vision and translation models, and is being evaluated in other scenarios, such as speech applications (Zhu & Gupta, 2018).

Model distillation. Another common technique to reduce model size, is model distillation (Ba & Caruana, 2014; Hinton et al., 2015; Polino et al., 2018). Distillation reduces the model size by training a smaller model (student model) that learns to match the output of a larger model (teacher model), given the same inputs. After training, the student model captures almost the same information as the teacher model but with fewer parameters.

2.2 Adversarial Attacks Against DL Models

A plethora of research is devoted to creating adversarial attacks against DL models (Shan et al., 2020; Nasr et al., 2021; Lovisotto et al., 2021; Hussain et al., 2021; Wu et al., 2021; Mao et al., 2019; Gu & Rigazio, 2014; Goodfellow et al., 2015; Carlini & Wagner, 2017b; Papernot et al., 2016c; Carlini & Wagner, 2017a; Papernot et al., 2016b; Tramèr et al., 2018). The basic idea is to find a small, human-imperceptible perturbation under some bound (e.g., ℓ_∞-norm), such that when the perturbation is added to the input sample, the model mispredicts. Depending on the underlying threat model, adversarial attacks can be whitebox (attackers have full access to model parameters) or blackbox (attackers have no access to model parameters).

Whitebox attacks. In a whitebox attack setting, attackers have access to the model's parameters. An attacker can use those to create a perturbation that maximizes the following loss function for a particular input and label:

$$A = \arg\max_{A \leq \epsilon} L(\theta, x + A, y) \tag{1}$$

where L is the loss function, θ the model parameters, and A an adversarial perturbation constructed for input x with label y under perturbation bound $A \leq \epsilon$.

While Equation 1 can be solved using an expensive method (box-constrained L-BFGS) (Szegedy et al., 2014), a much more efficient method with linear complexity is the commonly-used Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), in which an input is distorted by adding small perturbations ϵ with opposite signs to the gradients. By making the signs of the input perturbation opposite to the gradient, each dimension of the input will enlarge the error by ϵ, and a high-dimensional input leads to a large accumulative error. Specifically, FGSM generates an adversarial sample $\hat{x}$ with (∇_x is derivative to x):

$$\hat{x} = x + \epsilon * \text{sign}\left(\nabla_x L\left(\theta, x, y\right)\right) \tag{2}$$

R+FGSM (Tramèr et al., 2018) is an enhancement to FGSM that adds random perturbations to the noise in addition to the gradient maximization. Projected Gradient Descent (PGD) (Madry et al., 2018) further improves the attack by iteratively maximizing the loss

$$\hat{x}_{t+1} = \text{Clip}_{x,\epsilon}\{\hat{x}_t + \alpha\, \text{sign}(\nabla_x L(\theta, x, y))\} \tag{3}$$

where $\text{Clip}_{x,\epsilon}$ is a projection function to the region defined by input x and perturbation bound ϵ in the input space and initially $\hat{x}_0 = x$. The idea in PGD is to convert the one-step FGSM attack into multiple steps. In step $t+1$, FGSM is applied to step t's adversarial sample $\hat{x}_t$ with a step perturbation size of α, and the result is projected back to the overall perturbation bound ϵ. PGD is thus determined by three parameters: the number of steps t, the step size α, and the perturbation strength ϵ. PGD is one of the state-of-the-art adversarial attacks, and we use it as the primary baseline in our evaluation (§5). We also evaluate other baseline methods: Momentum PGD (Dong et al., 2017) and CW (Carlini & Wagner, 2017b) attacks, which perform worse than PGD (§5.4).

Blackbox attacks. Unlike whitebox attacks which allow full access to the model parameters in order to calculate gradients, blackbox attacks allow no such access. Prior work has demonstrated successful attacks in blackbox settings by learning a surrogate model and then attacking it instead (Papernot et al., 2017). The insight is that adversarial attacks target fundamental weaknesses in the learning method, and are thus transferable from one model to another (Papernot et al., 2016a).

Bit-Flip attack. Another form of attack is altering the model in memory. The bit flip attack (Rakin et al., 2019) is a general parameter attack that targets the weights of quantized DNNs by bit-flipping their values in memory, based on the idea that resource-limited platforms normally lack effective data integrity check mechanism. This attack is largely orthogonal to the aforementioned attacks: it makes changes to the device's memory, while the later makes changes to its input during the attack phase. Such an attack can be mitigated by performing checksums. Moreover, attacks on

inputs can be targeted to a large group of devices, as they do not require device infiltration.

Attacks against adapted models. Prior work on Defensive Quantization applies the PGD attack to both the original and the adapted models (Lin et al., 2019) and shows that with a small perturbation bound, adapted models are more robust than their original counterparts. However, once the perturbation increases beyond a certain point, adapted models exhibit larger errors and are less robust.

2.3 Robust Training

Much recent work defends against adversarial attacks via robust training (Han et al., 2018; Hendrycks et al., 2019; Wang et al., 2019; Wong et al., 2020). Robust training is typically formulated as a minimax problem and optimizes the model parameters to minimize the maximum loss an attack can induce (Madry et al., 2018; Lanckriet et al., 2002):

$$\min_{\theta} \max_{A \leq \epsilon} L(\theta, x + A, y) \tag{4}$$

This minimax method is considered a principled approach with roots from robust optimization (Scarf, 1958). It has been shown to boost model accuracy under attack by 28.6% (Michel et al., 2021), achieving the state of the art. Robust training is more expensive than regular training because of the high cost of solving the minimax as opposed to the minimization problem. Therefore it is typically only applied to the original full-precision models on powerful servers.

2.4 Differential Testing

Our work on exploiting the differences between two versions of a model is inspired by differential testing, a popular software testing method that detects the deviations between two implementations of the same functionality. It feeds the same input to the two implementations, observes the differences in the executions, and attempts to mutate the input to cause to larger differences. It has been shown to effectively detect many semantic bugs including SSL certification verification vulnerabilities (Chen & Su, 2015). The power of differential testing lies in that it cross-checks two implementations, using each as the reference for the other. Compared to the vast space of possible execution behaviors of one implementation, the deviations of the two implementations form a much smaller space.

A few prior systems applied differential testing to DL models (Pei et al., 2017; Guo et al., 2018; 2020). For instance, DeepXplore (Pei et al., 2017) generates adversarial samples with physically-realizable transformations (e.g., brightening and rotation) by finding deviations of two DL models. It formulates the problem as a joint optimization with several constraints, including that the perturbations must map to one of the transformations, and that the number of activated neurons is maximized. Like these systems, we employ the idea of differential testing but apply it to a different goal of differentially-attacking the adapted model, while referencing the original model.

3 Motivation

In this section we provide further motivation for DIVA. We describe the attack scenario, and then motivate DIVA using the example of quantization, a common edge-adaptation technique. We demonstrate that while quantization has a small impact on average accuracy, it causes bigger deviations on individual inputs. Finally, we demonstrate that existing attacks against quantized models also significantly affect the prediction of the original model, and thus would be easier to detect and debug when the original model undergoes validation and debugging.

Attack scenario. Companies that deploy models on edge devices need to deal with the huge diversity in hardware on different phones, tablets and cameras. For example, Facebook estimates that devices running its deep learning models comprise over 2000 unique SoCs running on tens of thousands of tablet and phone models. To accommodate such a large set of devices, for each one of their full-precision models, ML operators may need to create thousands of edge-adapted models versions (Wu et al., 2019). Even worse, since devices come online sporadically, some of the devices may still be running older versions of the adapted models. Due to the very large number of adapted models, and their similarity in top-line accuracy to the original model, ML operators typically only use the original version of the model for validation and debugging, which is also the case in popular machine learning frameworks (Abadi et al., 2016; tfl, 2021; pyt, 2021).

Therefore, an attack that only targets a particular edge model, but not the original model, would be harder to detect by the operator, and even if it was detected, it would be expensive to track down and debug. For example, in order to understand which devices and which model versions may be vulnerable to a particular set of adversarial inputs, the ML operator might have to sift through many adapted models or, worse, re-generate all the adapted models for all model versions on all possible edge devices, and manually run inference for each one of these models on the adversarial inputs. This process is time-consuming and expensive, and before the operator pinpoints the root cause of the attack, many edge devices may have been subverted.

We now provide motivation for how to construct such attacks using the example of quantization.

The impact of quantization on accuracy. In general, quantization has a relatively small impact on the *average accuracy*. Table 1 compares the average accuracy of the

Architecture	Original Accuracy	Quantized Accuracy	Original Correct & Quantized Incorrect	Original Incorrect & Quantized Correct	Total Instability
ResNet50	72.1%	70.1%	1510	925	8.1%
MobileNet	69.1%	67.4%	1199	677	6.3%
DenseNet121	73.5%	71.0%	1567	816	7.9%

Table 1. Comparing accuracy between the original full-precision models and quantized models, and the number of samples that deviate in both models. Instability is the total percentage of images that deviate between the models.

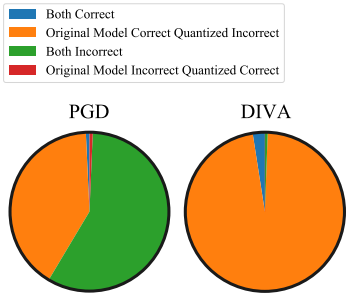


Figure 1. Results of the PGD and DIVA attacks on the original full-precision ResNet50 and its quantized version. PGD causes both versions of the model to misclassify. On the other hand, DIVA causes a misclassification only on the adapted version and mostly does not affect the original version.

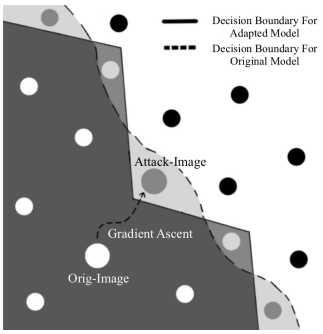


Figure 2. The decision boundaries of the adapted model in the input space are coarser-grained than those of the original model. DIVA leverages differential testing to locate the small deviations along the boundaries.

original floating-point model (fp32) to a model that uses quantization-aware training (int8 precision) across three different architectures. Across all three architectures, the accuracy achieved by the quantized model is 96% or more of the original model's accuracy.

While quantization does not significantly affect the top-line average accuracy, it introduces a subtle variance in prediction for individual inputs. The table shows the number of deviations in predictions, where one model predicts correctly and the other incorrectly. We also calculate the instability (Cidon et al., 2021) across the two versions – the percentage of total images where the models disagree. Across all three architectures, the instability varies between 6.3–8.1%, which is higher than the top-line accuracy metric on its own would suggest.

Baseline causes full-precision models to mispredict. Adversarial attacks have been shown to be transferable from one model to another (Papernot et al., 2016a). In other words, an attack that affects one model is likely to impact another. This means that if an attacker targets a quantized model using a standard attack, it is likely to also affect the original model. Indeed, Figure 1 shows the percent of images that get classified correctly and incorrectly by the original model and the quantized one when we apply PGD on a quantized ResNet50. The results show that PGD also effects the original model. Thus, since validation is typically conducted on the original model, it is likelier that such an attack would be detected. On the other hand, we will show that DIVA (§4), successfully attacks the quantized model, but it has a much smaller effect on the original model, and so it is likely to go undetected during validation.

4 Differential Evasive Attack

This section describes the construction of DIVA, starting from the intuition (§4.1), followed by the whitebox (§4.2), semi-blackbox (§4.3) and blackbox attacks (§4.4).

4.1 Intuition for DIVA

DIVA targets the deviation of the adapted model from the original model. The goal of DIVA is to be stealthy, because the adversarial samples that it generates mislead only the adapted model but otherwise behave correctly on the original model, making it challenging for the machine learning engineer to detect the attack during training and validation, which is typically conducted on a server or cluster setting. Even when detected, DIVA exposes vulnerabilities that are sticky and difficult to debug and remedy, because virtually all existing robust training techniques target original models and cannot directly apply to adapted models. While robust training may conceivably be accommodated to work with model adaptation, the model parameters are eventually compressed, "coarsening" the decision boundaries in the input space. The coarse-grained decision boundaries of the adapted model always deviate slightly from the fine-grained decision boundaries of the original model. However subtle the deviations may be, DIVA is designed to effectively locate them and exploit them to launch successful attacks.

Figure 2 shows how DIVA uses differential testing to locate the subtle deviations. Given a sample, DIVA observes how each of the two models compute the label for the sample. It then calculates a small perturbation that keeps the original model's label unchanged but reduces the probability for the adapted model to predict the same label. DIVA can jointly consider both models' probabilities and calculate the input perturbation accordingly. It repeats this step multiple times until it reaches one of the regions in the input space where the adapted and original models deviate.

4.2 Whitebox DIVA Attack

In the whitebox DIVA attack, the attacker has full access to both the original and adapted models. Similar to the general whitebox adversarial attack, which we described in Equation 1, the attacker can generate adversarial samples by solving an optimization problem that generates additive noise that maximizes the loss function. Unlike a standard whitebox attack, in the case of DIVA the loss function must jointly consider both the adapted and the original models, as shown in the following equation:

$$L_{DIVA}(\theta, x, y) = \text{model}_{orig}(x)[y] - c \cdot \text{model}_a(x)[y] \quad (5)$$

Here $\text{model}_{orig}(x)[y]$ is the raw probability of input x for label y in the original model's prediction, and $\text{model}_a(x)[y]$ is that for the adapted model. Hyperparameter c balances the two probabilities, and is set by default to $c = 1$. c represents a trade off between how well DIVA evades the original model and how well it attacks the adapted model. We further study the effect of c in §5.3. Loss function L_{DIVA} captures the difference between the raw probabilities of the two models. θ may refer to the parameters of the original model, the adapted model, or both, depending on how this loss function is used. Plugging this loss function into Equation 1 yields the following joint maximization problem:

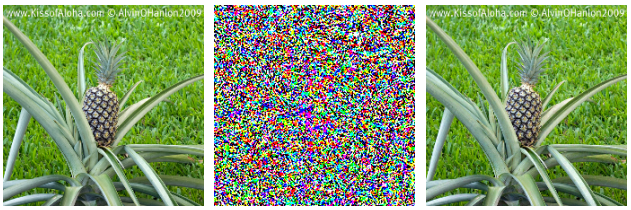
$$A = \arg\max_{A \leq \epsilon} L_{DIVA}(\theta, x + A, y) \quad (6)$$

that can be solved using stochastic gradient descent or other standard optimization techniques. Our design uses PGD to solve the problem and generate adversarial perturbations.

Example. We highlight an example of how DIVA can be used to trick an adapted model, depicted in Figure 3. In this example, the adversarial noise (3b) is added to an image (3a), to produce an attacked image (3c), which is indistinguishable to the human eye from the original image. The attacked image does not affect the prediction of an original ResNet50, which accurately predicts the object as a pineapple, but causes adapted model (TensorFlow, int8), running on a resource constrained device, to mispredict the image as a cairn (man-made pile of stones).

How does the DIVA adversarial noise affect the learned representations? To better understand how DIVA's generated adversarial noise affects both the original and adapted models, Figure 4 presents the Principal Component Analysis (PCA) (Bair et al., 2006) of the representations learned by ResNet50 on 1,000 samples that both the original and adapted models classify as digit 0 and another 1,000 that both classify as digit 2 from the MNIST hand-written digits dataset (LeCun et al., 1998)[1]. We pick the activations from the penultimate layer of ResNet50 as the representation of a sample. We similarly analyze the representations after the 1,000 digit 0 images are attacked by DIVA, which causes the adapted model to mispredict digit 2 while the original model still predicts correctly.

There are several interesting details in the PCA figure. First, even on the original images, there is a subtle difference in the representations learned by the adapted and the original model, in particular on digit 0 (the blue and orange dots in the upper part of the graph, which belong to the adapted and original models respectively). Second, the figure shows



(a) Original image. (b) Attack noise. (c) Attacked image.

Figure 3. Attack against an adapted model running on edge device that is undetected by the original model. With DIVA, a pineapple is identified as a cairn by the adapted model (confidence 79.2%), whereas the original model still recognizes the image as a pineapple (confidence 76.1%). The original image is recognized by both the original and the adapted models as a pineapple (confidence 100.0% and 98.5%, respectively).

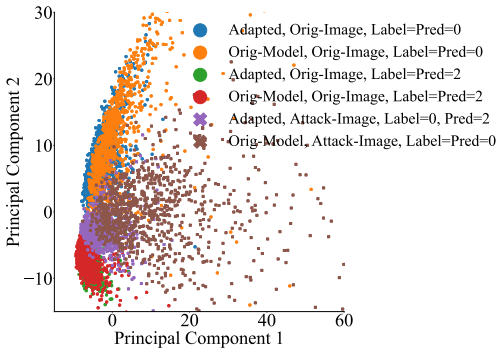


Figure 4. PCA visualization of representations learned by the original and adapted models, using the two most-principal components on MNIST. "Adapted, Orig-Image, Label=Pred=0" indicates the representations learned by the adapted model on the digit '0' original images. "Attack-Image" shows the representations on the images with DIVA's adversarial noise.

[1] We use MNIST for PCA visualization because each digit has many samples.

how DIVA shifts the representations both for the adapted and original models. The purple crosses show how DIVA is able to shift the images belonging to digit 0, from the upper part of the graph to the bottom part. For the original model, DIVA also shifts the points more towards to the bottom of the graph, but less so than it does for the adapted model, thus preserving the prediction of the correct label for most of them.

4.3 Semi-blackbox DIVA Attack

In the semi-blackbox attack, the attacker has access to the adapted model but not the original model. We assume it is substantially easier for the attacker to obtain one of the edge devices with the adapted model than hacking into a server with the original model. Practically, an attacker can obtain the adapted model from an edge device and recover the differentiable quantization model by extracting the zero points, scales and weights for each layer in the downloaded model, and retain its accuracy without any fine-tuning. However, without access to the original model parameters and training data, the gradient calculations in whitebox attack no longer apply.

Fortunately, attacks are frequently transferable (Papernot et al., 2016a): adversarial samples generated for one model often transfer to another model for a similar task. To launch a semi-blackbox attack, we reconstruct a full-precision surrogate model that behaves similarly as the original model and apply L_{DIVA} to generate adversarial samples for the adapted and the surrogate model. Figure 5 shows the semi-blackbox attack pipeline.

We reconstruct a surrogate model as follows. Based on the adapted model, we create an architecture that matches the adapted model. The surrogate model's parameters are initialized using the pretrained ImageNet parameters from TensorFlow(ker, 2020) when possible or the parameters of the adapted model.We then finetune the surrogate model via knowledge distillation. Unlike typical knowledge distillation that trains a model with less precision using an original model, we use knowledge distillation to create semi-blackbox attack. In our case, DIVA treats the adapted model as the teacher and the surrogate model as the student. We train the surrogate model to minimize the loss between its prediction and the label predicated by the adapted model while minimizing the distillation loss (Hinton et al., 2015).

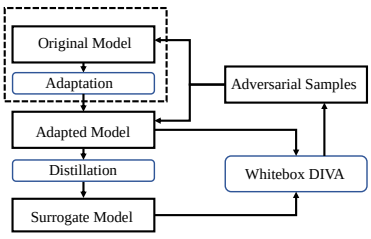


Figure 5. DIVA semi-blackbox attack pipeline. The dotted box indicates components that the attacker has no access to.

4.4 Blackbox DIVA Attack

In the full blackbox setting, the attacker does not have access to neither the entire full-precision model nor the parameters of the adapted model. A fully-blackbox attack can be generated by reconstructing both a surrogate full-precision model, described in §4.3, and a surrogate adapted model followed by adaptation and fine-tuning. The blackbox attacks then can be generated using the surrogate models and evaluated on the original full-precision and original adapted model.

5 Evaluation

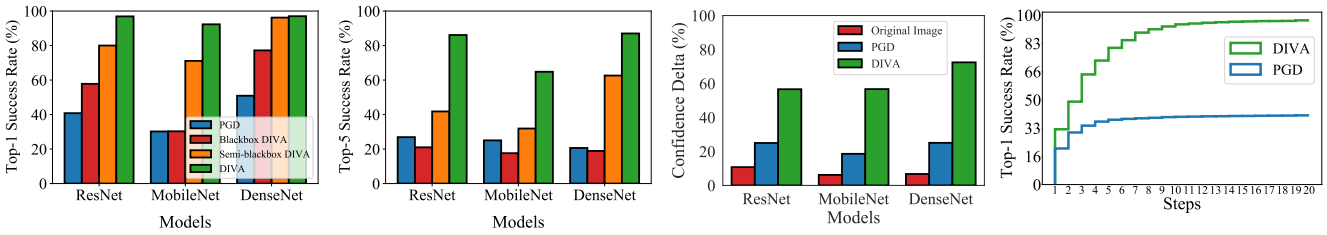
This section presents an evaluation of DIVA. It first describes the experimental setup (§5.1), then the efficacy of the DIVA whitebox, semi-blackbox and blackbox attacks (§5.2), and of a defense method (§5.5), and finally show that DIVA generalizes to pruning (§5.6).

5.1 Experimental Setup

Datasets. Our main dataset contains 50,000 images from the ImageNet Object Localization Challenge of 2012-2017 (ima, 2017). We randomly select 40% of the 50,000, or 20,000, images from the dataset as our training dataset. For the validation datasets, we randomly select 3,000 images from the remaining 30,000 images covering all 1,000 classes of ImageNet, with an average of three images per-class. When selecting these 3,000 validation images, we ensure that they are correctly classified by all relevant models and architectures. Eventually we created three sets of 3,000 validation images: one dataset for quantized whitebox/semi-blackbox attacks, one for quantized blackbox attack, and one for the attack on pruned models.

The semi-blackbox and blackbox attacks also require additional data to train the surrogate models. Therefore, we use 1% (12,811) images from the training dataset of the same ImageNet challenge to train the surrogate models. Since the semi-blackbox and blackbox attacks assume that the attacker cannot access any training data of the original model, we ensure that the 12,811 images have no overlap with our main dataset by selecting the 50,000 images in our main dataset from the validation dataset of the ImageNet challenge.

Models. We use three architectures in the evaluation: ResNet50, MobileNet and DenseNet121. For each one, we create four models: the original model, the adapted model, the surrogate original model for the semi-blackbox attack and the surrogate adapted model for the blackbox attack. We create the original models by downloading pre-trained models from TensorFlow Keras Applications (ker, 2020) and finetuning them on our training dataset of 20,000 images. We use pre-trained models instead of training from scratch to save resources and ensure that the models evalu-



(a) Top-1 success rate. (b) Top-5 success rate. (c) Confidence delta for top-1. (d) Top-1, varying attack steps.

Figure 6. **Attacks on quantized models.**

ated have state-of-the-art accuracy. The finetuning step is necessary for model adaptation, in the case when the dataset used for finetuning is a subset of the training dataset, not a disjoint dataset. We validate that the generated original models achieve the state-of-the-art accuracy.

We generate quantized models by first applying TensorFlow Model Optimization `tfmot`'s `quantize_model` (qua, 2020) on the original models using int8 quantization. We then apply QAT to these models on our training dataset of 20,000 images. After two epochs of QAT, the quantized models reach validation accuracy comparable to their corresponding original models. We observe that more epochs do not improve accuracy but worsen the stability between the original and the adapted models. The surrogate models are created by applying knowledge distillation and QAT with the aforementioned 12,811 images.

We generate two types of pruned models on the three architectures: (1) by applying Keras weight pruning on original models, and (2) by taking the pruned models and then quantizing them using the same pipeline we used for quantized models, while preserving the sparsity using `tfmot`. Both types of models are then fine-tuned to reach their highest accuracy on the dataset. After pruning, the model sizes were compressed to one third of their original size.

Attack construction. In our experiments, we construct the PGD attack targeting the adapted model as the baseline attack. We choose $\epsilon = 8$ for both baseline and DIVA attack, as it has been shown that the perturbation of $8/255$ is generally imperceptible to human eyes. We run both the baseline and the DIVA attacks with a step size $\alpha = 1$ (Lin et al., 2019), and set the maximum number of steps $t = 20$. We do not initialize the attack using random noise because random start is less effective in a single run. Instead, we initialize with a natural sample.

Success metrics. We define a successful attack as one where both: (a) the attack did not cause the original model to mispredict, i.e., if it correctly classified the original image, it correctly classifies the perturbed one; and (b) the attack caused the adapted model to mispredict, i.e., the adapted model correctly classified the natural image, but incorrectly classified the perturbed one. We define three metrics to quantify DIVA's efficacy. The first, which we refer to as *top-1 success rate*, measures whether the attack caused the top-1 prediction of the adapted model to be incorrect. The second, *top-5 success rate*, measures whether the attack caused the adapted model's top-1 prediction not to even appear in the top-5 predicted classes of the original model.

We define a third metric, *confidence delta*, which measures the difference between the confidence of the correct class between the original model and the adapted model, on the attacked image. For example, if the correct class in an image is a pineapple, and the confidence of the original model on the attacked image is 80%, while the confidence of the adapted model on the attacked image is 20%, the confidence delta will be equal to 60%. We additionally measure DSSIM (Hore & Ziou, 2010), which quantifies the similarity between two pictures from human's perspective, to examine whether the adversarial samples are similar to their corresponding natural samples.

Machine. Experiments are conducted on a server with 4 Intel 20-core Xeon 6230 CPUs, 376 GB of RAM, and 8 Nvidia GeForce RTX 2080 Ti GPUs with 11 GB of memory.

5.2 DIVA with Quantization

DSSIM. We measure DSSIM among all generated adversarial images on three model architectures comparing the PGD baseline against DIVA. The resulting DSSIM for all images are below 0.0092, which means that both the PGD and DIVA attacks are imperceptible to humans.

Top-1 and Top-5. Figure 6 compares the performance of DIVA with PGD, under the three architectures. Figure 6a shows that DIVA significantly outperforms the baseline attack method, PGD, both for the whitebox and semi-blackbox scenarios. The whitebox attack is able to cause the adapted model to misclassify the correct label, while maintaining the original model's correct classification, for 92.3–97% of the validation dataset. PGD is also able to successfully attack some of images, because in general the adapted model is less robust to noise than the original model. However, its attack success rate is much lower than DIVA: it achieves a 30.2–50.9% top-1 success rate. As illustrated previously in Figure 1, while the PGD baseline is able to cause the

adapted model to mispredict, the adversarial samples that it generates also cause the original model to mispredict, a key reason that it is substantially less effective than DIVA.

Figure 6d plots the top-1 success rates for PGD and DIVA on ResNet50 as the number of attack steps increases. DIVA outperforms PGD starting from step 1. PGD plateaus at 40.8% on step 7, whereas DIVA reaches 96.9% on step 11. For the top-5 success rate, the differences are even more stark, as DIVA in the whitebox setting is able to achieve a 2.6-4.2× higher top-5 success rate than PGD. As expected, the semi-blackbox attack is less successful than the whitebox one, since DIVA does not have access to the original original model. However, it is still able to outperform PGD across all the experiments. It has a high success rate especially for the top-1 metric, successfully attacking 71.1-96.2% of the images across the three models.

Blackbox DIVA does not perform as well as the semi-blackbox version, resulting in a top-1 success rate of 30.3–77.2%, which is higher than PGD's. However the blackbox attack's top-5 success rate is 17.6–21%, which is lower than PGD's. Since the blackbox attack has to generate an extra surrogate model compared to the semi-blackbox version, it is harder for it to locate the decision boundaries. We further noticed that the top-1 success rate for blackbox DIVA on MobileNet is worse than the other two networks, when comparing the improvements with their PGD baselines. This is because small under-parameterized neural networks often generalize worse than larger networks (Novak et al., 2018), leading to poor transferabilty of DIVA from the surrogate models to original models.

Confidence delta. Figure 6c presents the average shift in confidence for the correct label under the three models, comparing DIVA (in a whitebox setting) to PGD. For each model, we first compare the average confidence delta of the original image, between the original model and the adapted one. As expected, in all three models, there is a relatively modest difference in the confidence delta solely due to quantization (on average only 7.9%). As expected, the attacks cause the confidence delta to shift more. For PGD, the average confidence delta varies between 18.6–25% across the three models. For DIVA, the average confidence delta varies between 56.6–72.4%. The much higher shift in confidence for the correct prediction explains why DIVA has a much higher top-1 and top-5 success rate than PGD.

Attack speed. We measure the wall-clock time for both PGD and DIVA with our experimental setup. They run at almost the same speed of one second per step.

Evasion cost. We measure the cost of evading the original model, by comparing the success of DIVA at attacking the adapted model compared to PGD. Note that we generate the adversarial samples using DIVA as usual, considering both the original and adapted models. However, to compute the success rate, the only criterion is that an adversarial sample misleads the adapted model. The results are presented in Table 2, and show that, despite the constraint not to affect the original models, DIVA achieves a very high attack success rate on the quantized models, but falls slightly short of PGD (1.7–3.6% less effective). When attacking the pruned models, which we describe in more detail in §5.6, DIVA is almost as effective as PGD and just falls negligibly short of PGD in few cases (0.2–0.4% less effective).

Architecture	PGD Attacking Quantized	DIVA Attacking Quantized
ResNet50	98.7%	97.0%
MobileNet	98.7%	95.1%
DenseNet121	98.4%	96.7%

Table 2. Comparing attack success rate solely against adapted models. For pruning, DIVA and PGD both achieve 100%. For pruning + quantization, PGD has success rates of 98.4–99.7% and DIVA 98–99.7%.

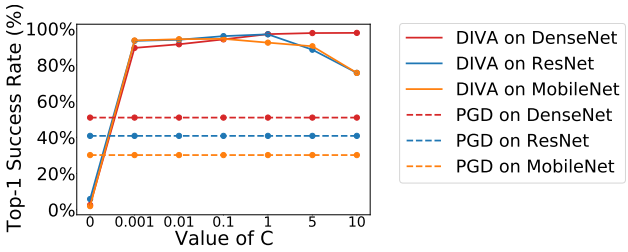


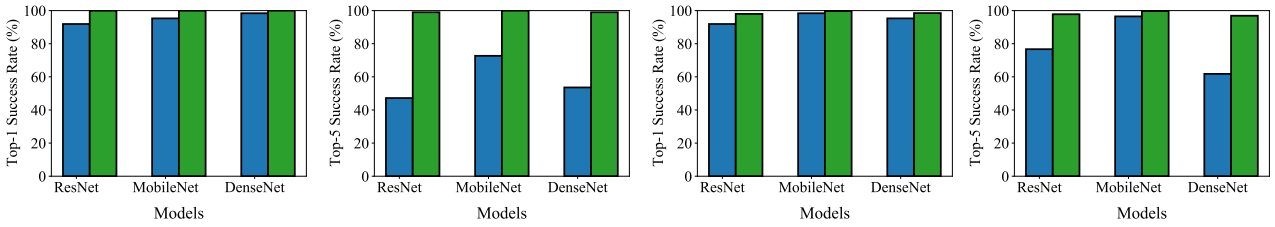
Figure 7. Whitebox DIVA with varying c.

5.3 Balancing Between Evading and Attacking

The hyper-parameter c is used to balance the effect of the two loss terms in Equation 5. A small c focuses the attack on not being detected by the original model, and a large c focuses on attacking the adapted model.

Under the setting of quantization, we conduct an ablation experiment for $c = \{0, 0.001, 0.1, 1, 5, 10\}$ with whitebox DIVA. Figure 7 shows the top-1 attack success rate reaches a peak at $\{96.9\%, 94.4\%, 97.7\%\}$ when $c = \{10, 1, 0.1\}$, respectively for each network we study. The results show that the attack achieves a relatively high success rate for the studied models for $c = [0.001, 1]$. We set $c = 1$ by default because it provides the highest average top-1 success across the three model architectures.

The success rate on solely attacking the adapted model is increased to $\{97.6\%, 98\%, 97.7\%\}$, respectfully for each model architecture when $c = 10$, compared to the numbers in Table 2 when $c = 1$. The results show that the evasion cost can be reduced by tuning c at the expense of a lower average evasion success rate.



(a) Top-1 success rate. (b) Top-5 success rate. (c) Top-1 success rate. (d) Top-5 success rate.

Figure 8. **Attacks on pruned models.** (a,b) Pruned models. (c,d) Pruned and quantized models.

5.4 Other Baseline Attacks

We also evaluated other baseline methods: the L_∞ CW attack (Carlini & Wagner, 2017b) and Momentum PGD (Dong et al., 2017) attack, under the quantization setting and the top-1 success criterion. We follow the same hyper-parameter setup as the CW attack in Madry et al (Madry et al., 2018). For Momentum PGD, we use a momentum term of 0.5 in addition to the standard PGD attack as this gives the best performance. Our results show that across the three architectures, CW achieved an average success rate of 25.5% and Momentum PGD achieved 39.4%, both of which are worse than PGD (40.6%).

5.5 Robust Training as a Defense

As explained in §2.3, robust training can effectively reduce the success rates of adversarial attacks such as PGD. This subsection compares the effectiveness of the PGD minimax robust training on both PGD and DIVA.

Since the original PGD robust training code (Madry et al., 2018; rob, 2021) is implemented in Pytorch, we ported DIVA to Pytorch for fair comparison. We used the default hyperparameters used by the authors (rob, 2021) ($\epsilon = 8/255 \approx 0.03$, an attack learning rate of 0.00375 with 20 attack steps without random start). We used the pre-trained robust Resnet50 model in the library as the original model and generated the corresponding adapted model using the PyTorch-Quantization toolkit (pyt, 2021).

Our results show that when we use $c = 5$ in DIVA, it improves the top-1 success rate from 10.5% to 12.8%, compared to PGD. Meanwhile, the evasion cost of DIVA is as similar as PGD, or even slightly better, resulting a robust accuracy of 22.63% for PGD, and 21.77% for DIVA on quantized model. We found the best c value for generating the evasive attack under robust training is $c = 1.5$. This value produces a 4% lower success rate on attacking the adapted model, but the overall evasive attack success rate is 10.1% higher compared to PGD. Both the evasive attacks' success rates for PGD and DIVA drop when they attack the robust trained models. We think this is because the non-overlapping area between the decision boundaries of the adapted model and the original model becomes smaller, due to the fact that they are both trained to cover the worst case attacks.

5.6 DIVA against Pruning Adaptation

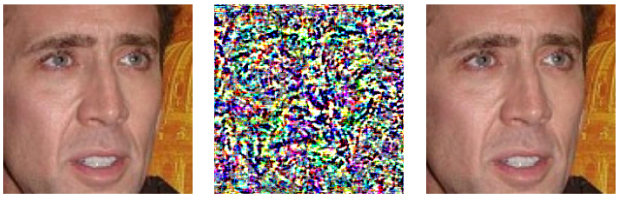
We evaluate whitebox DIVA on pruned models. Figure 8 shows that under both the top-1 and top-5 metrics, DIVA achieves a success rate of 97.8% or higher in all cases and always performs better than PGD. The top-1 success rates of PGD and DIVA are much closer than they are in the quantization setting potentially because pruning makes more intrusive changes to the model weights than quantization. The larger gap between the pruned and the original models, exhibited by the high instability ranging from 17.1–33.5% and the high confidence delta on the original images ranging from 10–36.1%, allows PGD to attack the pruned models without collaterally damaging the original models. However, DIVA increases the confidence delta on the perturbed images by 8.3–16% more than PGD. The top-5 success rates of DIVA in most settings are significantly higher than PGD.

6 Case Study: Face Recognition

We present the results of our case study that uses DIVA to attack a face recognition model, whose quantized version represents a model that would be running on an edge device (e.g., security camera or phone). For example, DIVA causes the model to misidentify Nicolas Cage as Jerry Seinfeld as shown in Figure 9.

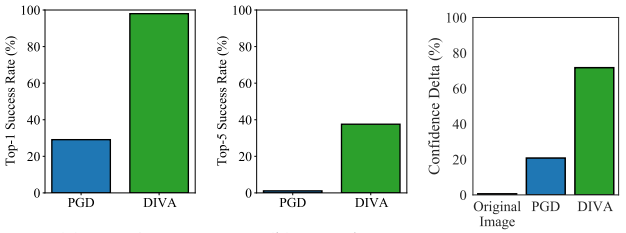
Dataset. Our dataset includes 11,640 images belonging to 150 people from the PubFig database (Kumar et al., 2009). We pick the first 90% of the images from the dataset to finetune our models. For the validation dataset, we randomly select 450 images from the remaining 10% of the dataset that both the full-precision and the adapted models classify correctly. The validation dataset covers all 150 labels of our dataset, with three images per class.

Models. We use the VGGFace model, which internally employs ResNet50's architecture, and evaluate it in the whitebox setting. The original model is constructed via finetuning a pre-trained model with initial full-precision parameters trained on the VGGFace2 dataset (Cao et al., 2017). We construct the QAT model by applying Tensor-



(a) Original image. (b) Attack noise. (c) Attacked image.

Figure 9. DIVA causes the adapted model to misidentify Nicolas Cage as Jerry Seinfeld with 95.7% confidence, whereas the full-precision model still recognizes the face as Nicolas Cage with 93.4% confidence. The original face is recognized by both models as Nicolas Cage with 100% confidence.



(a) Top-1 success rate. (b) Top-5 success rate. (c) Confidence delta.

Figure 10. Attacks on the face recognition model.

Flow tfmot's `quantize_model` (qua, 2020) on the original model. During QAT, we further finetune this model using the same dataset. Finally, we convert the QAT model to a real adapted int8 model with Tflite in order to evaluate it on a resource-constrained device with AArch64 CPU, which would represent the type of processor that might run on an edge device, such as an iPhone (arm, 2017). Since Tflite supports only inference and does not expose the gradients, we use QAT's gradients in constructing the DIVA attacks.

Evaluation. Figure 10 presents the results of the experiment. On the surface, the original and adapted models have very similar accuracy: the original model has an accuracy of 99.4%, while the adapted model has an accuracy of 99.0%. However, the high success rates of DIVA indicate that these models have many subtle differences. Similar to our results on ImageNet, DIVA significantly outperforms PGD with the face recognition task. DIVA is somewhat less successful in the top-5 metric compared to the ImageNet models, likely due to the fact that PubFig dataset contains only 150 classes (i.e., people), compared to the 1,000-class ImageNet dataset.

Targeted attack. We construct a simple targeted attack that attempts not only to fool the adapted model but also lead it to predict a particular person or set of people. This entails adding an additional loss term to our attack, which increases the loss based on its distance away from a one-hot vector with the value of 1 being at the position of the target class. We evaluated the attack on 10 people and were able to target the misclassification on average to a set of 8.3 people (out of the 150 people in the dataset). We believe this attack can be further fine-tuned and be made more accurate.

7 Other Related Work

We covered the closely related work in §2. In this section, we discuss more tangentially related work.

Quantization. Quantization of DNNs is a widely-used technique, in order to allow for efficient inference and to fit larger models on resource constrained edge-device hardware (Jacob et al., 2018; Cai et al., 2017; Han et al., 2016; Lin et al., 2016). This work mostly focuses on minimizing the accuracy degradation when compressing a model. Prior work has also shown that even when accuracy is maintained between source model and the quantized version, there is still a prediction divergence between them that can cause instability (Cidon et al., 2021). This forms the basis for DIVA's differential approach.

Model compression and robustness. Past work has studied the relationship between model compression and robustness (Lin et al., 2019; Ye et al., 2019; Gui et al., 2019; Stutz et al., 2021; Liebenwein et al., 2021). To the best of our knowledge, all past work focused on robustness to traditional adversarial attacks and random noise. Unlike past work, DIVA focuses on the prediction *divergence* between the original model and its quantized version.

8 Conclusions

The deployment of DL models in large-scale settings on tens of millions of edge devices creates new security vulnerabilities. This paper highlights a new differential attack, DIVA, which exploits the subtle differences between the edge-adapted versions of the models and the original server-based model. DIVA constructs adversarial noise that maximizes the loss of the edge model while minimally affecting the inference of the original model. This causes the edge model to mispredict while significantly increasing the cost of detection and debugging. We adapted this attack to a setting where the attacker only has access to the edge model but not the original model, and showed that it remains effective. We hope this work opens the door to a new line of research on attacks and defenses that target the variations in models deployed in production.

9 Acknowledgments

We thank the reviewers for their comments. This work was supported by a grant from the Columbia Center of AI Technology in collaboration with Amazon, ONR grants N00014-16-12263 and N00014-17-1-2788; NSF grants CNS-2104292, CNS-1750558 and CNS-1564055; a Facebook gift; a JP Morgan Faculty Research Award; and a DiDi Faculty Research Award.

REFERENCES

iOS Device Compatibility Reference . https://developer.apple.com/library/archive/documentation/DeviceInformation/Reference/iOSDeviceCompatibility/DeviceCompatibilityMatrix/DeviceCompatibilityMatrix.html, 2017.

An On-device Deep Neural Network for Face Detection . https://machinelearning.apple.com/research/face-detection, 2017.

ImageNet Object Localization Challenge. https://www.kaggle.com/c/imagenet-object-localization-challenge, 2017.

Module: tf.keras.applications . https://www.tensorflow.org/api_docs/python/tf/keras/applications, 2020.

tfmot quantize_model. https://www.tensorflow.org/model_optimization/api_docs/python/tfmot/quantization/keras/quantize_model, 2020.

TensorFlow mixed precision. https://www.tensorflow.org/guide/mixed_precision, 2021.

Floating point and IEEE 754 compliance for NVIDIA GPUs. https://docs.nvidia.com/cuda/floating-point/index.html, 2021.

PyTorch Quantization. https://pytorch.org/docs/stable/quantization.html, 2021.

Robustness Package. https://github.com/MadryLab/robustness, 2021.

TensorFlow Lite. https://www.tensorflow.org/lite/guide, 2021.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Ahmadyan, A. and Hou, T. Real-Time 3D Object Detection on Mobile Devices with MediaPipe. https://ai.googleblog.com/2020/03/real-time-3d-object-detection-on-mobile.html, 2020.

Ba, J. and Caruana, R. Do deep nets really need to be deep? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2654–2662, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/ea8fcd92d59581717e06eb187f10666d-Abstract.html.

Bair, E., Hastie, T., Paul, D., and Tibshirani, R. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137, 2006. doi: 10.1198/016214505000000628.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Cai, Z., He, X., Sun, J., and Vasconcelos, N. Deep learning with low precision by half-wave gaussian quantization. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5406–5414. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.574. URL https://doi.org/10.1109/CVPR.2017.574.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. Vggface2: A dataset for recognising faces across pose and age. *CoRR*, abs/1710.08092, 2017. URL http://arxiv.org/abs/1710.08092.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pp. 3–14, New York, NY, USA, 2017a. Association for Computing Machinery. ISBN 9781450352024. doi: 10.1145/3128572.3140444. URL https://doi.org/10.1145/3128572.3140444.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (S&P)*, pp. 39–57. IEEE, 2017b.

Chen, Y. and Su, Z. Guided differential testing of certificate validation in SSL/TLS implementations. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pp. 793–804, 2015.

Cidon, E., Pergament, E., Asgar, Z., Cidon, A., and Katti, S. Characterizing and taming model instability across edge devices. *Proceedings of Machine Learning and Systems*, 3, 2021.

Deng, L., Li, G., Han, S., Shi, L., and Xie, Y. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

Dong, Y., Liao, F., Pang, T., Hu, X., and Zhu, J. Discovering adversarial examples with momentum. *CoRR*, abs/1710.06081, 2017. URL http://arxiv.org/abs/1710.06081.

Good, O. How Google Translate squeezes deep learning onto a phone. https://ai.googleblog.com/2015/07/how-google-translate-squeezes-deep.html, 2015.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

Gu, S. and Rigazio, L. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

Gui, S., Wang, H., Yang, H., Yu, C., Wang, Z., and Liu, J. Model compression with adversarial robustness: A unified optimization framework. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1283–1294, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/2ca65f58e35d9ad45bf7f3ae5cfd08f1-Abstract.html.

Guo, J., Jiang, Y., Zhao, Y., Chen, Q., and Sun, J. Dlfuzz: differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 739–743, 2018.

Guo, J., Zhao, Y., Jiang, Y., and Song, H. Coverage guided differential adversarial testing of deep learning systems. *IEEE Transactions on Network Science and Engineering*, 2020.

Guo, P., Hu, B., and Hu, W. Mistify: Automating DNN model porting for on-device inference at the edge. In *NSDI*, pp. 705–719, 2021.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *arXiv preprint arXiv:1804.06872*, 2018.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1510.00149.

Hendrycks, D., Lee, K., and Mazeika, M. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pp. 2712–2721. PMLR, 2019.

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

Hore, A. and Ziou, D. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pp. 2366–2369. IEEE, 2010.

Hussain, S., Neekhara, P., Dubnov, S., McAuley, J., and Koushanfar, F. WaveGuard: Understanding and mitigating audio adversarial examples. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, August 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/hussain.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A. G., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 2704–2713. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00286. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html.

Kumar, N., Berg, A. C., Belhumeur, P. N., and Nayar, S. K. Attribute and simile classifiers for face verification. In *2009 IEEE 12th International Conference on Computer Vision*, pp. 365–372, 2009. doi: 10.1109/ICCV.2009.5459250.

Lanckriet, G. R., Ghaoui, L. E., Bhattacharyya, C., and Jordan, M. I. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3(Dec):555–582, 2002.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liebenwein, L., Baykal, C., Carter, B., Gifford, D., and Rus, D. Lost in pruning: The effects of pruning neural networks beyond test accuracy. *Proceedings of Machine Learning and Systems*, 3, 2021.

Lin, D. D., Talathi, S. S., and Annapureddy, V. S. Fixed point quantization of deep convolutional networks. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2849–2858. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/linb16.html.

Lin, J., Gan, C., and Han, S. Defensive quantization: When efficiency meets robustness. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryetZ20ctX.

Lovisotto, G., Turner, H., Sluganovic, I., Strohmeier, M., and Martinovic, I. SLAP: Improving physical adversarial examples with short-lived adversarial perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, August 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/lovisotto.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Mao, C., Zhong, Z., Yang, J., Vondrick, C., and Ray, B. Metric learning for adversarial robustness. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

McKeeman, W. M. Differential testing for software. *Digital Technical Journal*, 10(1):100–107, 1998.

Michel, P., Hashimoto, T., and Neubig, G. Modeling the second player in distributionally robust optimization. *CoRR*, abs/2103.10282, 2021. URL https://arxiv.org/abs/2103.10282.

Nasr, M., Bahramali, A., and Houmansadr, A. Defeating DNN-based traffic analysis systems in real-time with blind adversarial perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, August 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/nasr.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJC2SzZCW.

Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016a.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pp. 372–387. IEEE, 2016b.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, 2016c. doi: 10.1109/SP.2016.41.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Pei, K., Cao, Y., Yang, J., and Jana, S. DeepXplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, 2017.

Polino, A., Pascanu, R., and Alistarh, D. Model compression via distillation and quantization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1XolQbRW.

Rakin, A. S., He, Z., and Fan, D. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1211–1220, 2019.

Rocki, K., Essendelft, D. V., Sharapov, I., Schreiber, R., Morrison, M., Kibardin, V., Portnoy, A., Dietiker, J., Syamlal, M., and James, M. Fast stencil-code computation on a wafer-scale processor. In Cuicchi, C., Qualters, I., and Kramer, W. T. (eds.), *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, pp. 58. IEEE/ACM, 2020. doi: 10.1109/SC41405.2020.00062. URL https://doi.org/10.1109/SC41405.2020.00062.

Scarf, H. A min-max solution of an inventory problem. *Studies in the mathematical theory of inventory and production*, 1958.

Shan, S., Wenger, E., Wang, B., Li, B., Zheng, H., and Zhao, B. Y. Gotta catch'em all: Using honeypots to catch adversarial attacks on neural networks. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 67–83, 2020.

Stutz, D., Chandramoorthy, N., Hein, M., and Schiele, B. Bit error robustness for energy-efficient dnn accelerators. In *Fourth Conference on Machine Learning and Systems*. mlsys. org, 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I. J., Boneh, D., and McDaniel, P. D. Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rkZvSe-RZ.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 322–330, 2019.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Wu, C.-J., Brooks, D., Chen, K., Chen, D., Choudhury, S., Dukhan, M., Hazelwood, K., Isaac, E., Jia, Y., Jia, B., et al. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 331–344. IEEE, 2019.

Wu, X., Guo, W., Wei, H., and Xing, X. Adversarial policy training against deep reinforcement learning. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, August 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/wu-xian.

Ye, S., Xu, K., Liu, S., Cheng, H., Lambrechts, J.-H., Zhang, H., Zhou, A., Ma, K., Wang, Y., and Lin, X. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Zhu, M. and Gupta, S. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Sy1iIDkPM.

A ARTIFACT APPENDIX

A.1 Abstract

This artifact appendix helps readers to reproduce the main experimental results in this paper. In this artifact evaluation, we show (1) how to create dataset and models for evasive attack evaluation. (2) how to perform DIVA in white-box, semi-blackbox and blackbox setting and perform PGD, Momentum PGD and CW attack as baselines. (3) how to perform analysis on the attack results.

A.2 Artifact check-list (meta-information)

- **Model:** ResNet50, DenseNet121, MobileNet, VGGFACE
- **Data set:** ImageNet2012, PubFig, MNIST
- **Run-time environment:** Debian GNU/Linux (or equivalent)
- **Hardware:** see §A.3.2
- **Metrics:** top-1, top-5 success rate, confidence score delta
- **How much disk space required (approximately)?:** 15MB (code only)
- **How much time is needed to prepare workflow using single GPU (approximately)?:** Model generation: 24hrs for 15 models that needs training; Dataset: 6hrs
- **How much time is needed to complete experiments using single GPU (approximately)?:** Attack: 3.5hrs each in pruning setting, 10 hrs each in quantization setting; evaluation for one attack: 10 hrs
- **Publicly available?:** Yes
- **Code licenses (if publicly available)?:** MIT
- **Workflow framework used?:** see §A.3.3
- **Archived (provide DOI)?:** 10.5281/zenodo.6084154

A.3 Description

A.3.1 How to access

- Github repository (https://github.com/WeiHao97/DIVA) contains latest codes to reproduce all experiment results
- Zip file (https://drive.google.com/file/d/1jy7AVFU8v8lbt8rcTWabV5-WLg9BjRcd/view) contains code, weights and datasets that can be run independently to reproduce the results from the case study (section 6) in this paper.

A.3.2 Hardware dependencies

- All experiments on 'server' are conducted on a server with four Intel 20-core Xeon 6230 CPUs, 376 GB of RAM, and eight Nvidia GeForce RTX 2080 Ti GPUs each with 11 GB memory.
- All experiments on 'edge' are conducted on a cloudlab m400 machine with eight 64-bit ARMv8 (Atlas/A57) cores CPUs, 62GiB of RAM. The machine's profile is 'ubuntu18-arm64-retrowrite-CI-2' running on node ms0633 in Utah (https://www.cloudlab.us/).

A.3.3 Software dependencies

- On server: jupyter notebook, numpy 1.19.5, tensorflow 2.4.1, keras 2.4.3, tensorflowmodeloptimization, kerasvggface , matplotlib, livelossplot, spicy, PIL, tensorflow_datasets, scikit-learn, seaborn, pandas,torch, torchvision, GPUtil, dill, tensorboardX, tables, dssim
- On edge: jupyter notebook, tflite-runtime

A.3.4 Datasets

We employ ImageNet, MNIST and PubFig in our experiments. PubFig is included in the zip file. ImageNet2012 has to be download manually from https://image-net.org/challenges/LSVRC/2012/, the code parses it automatically. MNIST is automatically loaded from TensorFlow Datasets by the code.

A.4 Installation

All software dependencies can be installed using pip.
dssim package for image similarity analysis can be downloaded from: https://github.com/kornelski/dssim

A.5 Experiment workflow

A.5.1 Quantization Experiments (§5.2–§5.4)

First, generate original full-precision, quantized and surrogate full-precision, quantized models by following the workflow in quantization/model_generate*.ipynb. Next, generate the 3000 images datasets for attack evaluation by following quantization/-generateImagePerClass.ipynb. Then, generate the DIVA and baseline attacks by running quantization/*.py. See quantization/README.md for more details.

A.5.2 Pruning Experiments (§5.6)

First, generate pruned models by following the workflow in pruning/ModelGen.ipynb. Next, generate the 3000 images dataset for attack evaluation by following pruning/generateImagePerClass.ipynb. Notice that this dataset is different from the datasets in §A.5.1. Then, run attacking scripts under pruning/attacks. See pruning/README.md for more details.

A.5.3 Case Study Experiments (§6)

Extract the zip file which contains code, weights, and data that can be run independently to reproduce the result of this section.

- For untargeted attack: First, split PubFig dataset for model training. Next, construct full-precision, QAT and tflite models and create the attacking dataset following FR_server.ipynb on the server. Then generate attacks following untargetted/*.py scripts the server and load the results on the edge. On the edge, run FR_edge.ipynb to evaluate the final top-1/top-5 success rate and the confidence delta on the generated

results.
- For targeted attack, run targetted.py to generate a dictionary where its key is the person's name and its value is an array of successful attacks.

See DIVA/quantization/PubFig/README.md for more details

A.5.4 Attacks under Robust Model Experiments (§2.3)

First, load the robust-trained full-precision model from https://github.com/MadryLab/robustness and create the robust-trained quantized model. Then, generate the attack using PGD and DIVA functions on ImageNet Dataset. The whole workflow including evaluation is included in robustness/notebooks/DIVA_under_robust_trained model.ipynb.
See robustness/README.md for more details.

A.5.5 MNIST Experiments (Fig 4)

First, train model on the MNIST dataset by following the workflow in quantization/Mnist/ModelGen.ipynb. Next, run DIVA attack by following the workflow in quantization/Mnist/attacks.ipynb. Last, generate visualization for attack results by running quantization/Mnist/PCA_TSNE.ipynb. See quantization/Mnist/README.md for more details.

A.6 Evaluation and expected result

The top-1/top-5 success rates for section (§A.5.1–A.5.4) can be found both in stdout and in the evaluation script for each experiment. The confidence deltas, DSSIM and evasion cost analysis are calculated in each evaluation script if they are evaluated in the paper.
The top-1/top-5 success rates for §A.5.4 can be found in stdout with the format:

- Total: {} Success: {} Q_W:{} FP_W:{} Robust_acc: {}

success/total gives the success rate and Robust_acc gives the robustness accuracy evaluated in the paper. Q_W and FP_W gives the number of mis-predictions by the full-precision and the quantized model after attack.

A.7 Experiment customization

Hyper-parameters can be customized:

- c: balancing the effect of attack on full-precision and quantized model.
- $grad\_iterations$: number of attack steps
- $step$: step size of each attack step
- $epsilon$: bound for the attack
- number of training steps during model generation.

A.8 Methodology

Submission, reviewing and badging methodology:

- http://cTuning.org/ae/submission-20200102.html
- http://cTuning.org/ae/reviewing-20200102.html
- https://www.acm.org/publications/policies/artifact-review-badging