Exploiting Edge Features for Transferable Adversarial Attacks in Distributed Machine Learning

Giulio Rossolini[a,*], Fabio Brau[b], Alessandro Biondi[a], Battista Biggio[b], Giorgio Buttazzo[a]

[a]*Department of Excellence in Robotics and AI, TeCIP, Scuola Superiore Sant'Anna, Piazza Martiri della Libertà 33, 56127, Pisa, Italy*
[b]*Department of Electrical and Electronic Engineering, University of Cagliari, Via Marengo 2, Cagliari, 09123, Italy*

Abstract

As machine learning models become increasingly deployed across the edge of Internet of Things environments, a partitioned deep learning paradigm in which models are split across multiple computational nodes introduces a new dimension of security risk. Unlike traditional inference setups, these distributed pipelines span the model computation across heterogeneous nodes and communication layers, thereby exposing a broader attack surface to potential adversaries. Building on these motivations, this work explores a previously overlooked vulnerability: even when both the edge and cloud components of the model are inaccessible (i.e., black-box), an adversary who intercepts the intermediate features transmitted between them can still pose a serious threat. We demonstrate that, under these mild and realistic assumptions, an attacker can craft highly transferable proxy models, making the entire deep learning system significantly more vulnerable to evasion attacks. In particular, the intercepted features can be effectively analyzed and leveraged to distill surrogate models capable of crafting highly transferable adversarial examples against the target model. To this end, we propose an exploitation strategy specifically designed for distributed settings, which involves reconstructing the original tensor shape from vectorized transmitted features using simple statistical analysis, and adapting surrogate architectures accordingly to enable effective feature distillation.

A comprehensive and systematic experimental evaluation has been conducted to demonstrate that surrogate models trained with the proposed strategy, i.e., leveraging intermediate features, tremendously improve the transferability of adversarial attacks. These findings underscore the urgent need to account for intermediate feature leakage in the design of secure distributed deep learning systems, particularly in edge scenarios, where constrained devices are more exposed to communication vulnerabilities and offer limited protection mechanisms.

Keywords: Distributed DNNs, Secure AI, Adversarial Transferability, Secure IoT, Black-Box Attack, Feature Distillation

1. Introduction

The growing adoption of deep learning across a wide range of application domains, including internet of things (IoT) and edge artificial intelligence (AI), has led to an increasing interest in partitioned learning and inference, paradigms in which a deep neural network (DNN) is divided across multiple heterogeneous nodes [1, 2, 3, 4, 5]. In a standard vertical partitioning scheme, the setting addressed in this work, the initial layers of the model are executed locally on the edge device, while the remaining layers are offloaded to the cloud.

While the partitioned paradigm offers clear advantages in terms of efficiency and scalability, it also introduces new security vulnerabilities. In the broader context of secure and robust AI, deep learning models are known to be susceptible to evasion attacks [6, 7], in which adversaries craft subtle perturbations to input data that lead to incorrect predictions. Although a large part of the literature focuses on white-box scenarios, where the attacker has full access to model parameters and gradients, such assumptions are often unrealistic in practice.

More practical threat models, in fact, adopt a black-box setting, in which the attacker has no knowledge of the architecture and inner parameters of the target model and can only rely on the outcome on known samples through a query-feedback mechanism. Under these constraints, a common procedure involves first training a *surrogate* network to approximate the behavior of the *target* model and then crafting adversarial examples on this surrogate in the hope that they transfer to the target [8, 9, 10]. In this setting, we argue that partitioned inference introduces a broader and largely underexplored adversarial surface: in addition to inputs and outputs, partitioned models transmit intermediate features across computing nodes, which may be intercepted in unprotected or misconfigured environments, as illustrated in Figure 1. This represents a realistic threat in resource-constrained IoT deployments [11, 12]. In fact, such intercepted features can be used to train more accurate surrogate models, significantly enhancing the effectiveness of transferrable attacks, as further discussed in Section 2.

This work investigates the vulnerabilities introduced in parti-

*Corresponding author
Email addresses: giulio.rossolini@santannapisa.it (Giulio Rossolini), fabio.brau@unica.it (Fabio Brau), alessandro.biodni@santannapisa.it (Alessandro Biondi), battista.biggio@unica.it (Battista Biggio), giorgio.buttazzo@santannapisa.it (Giorgio Buttazzo)

tioned models by showing how intermediate feature leakage in partitioned inference can be exploited to train surrogates suitable for generating more transferable attacks than with previous techniques for non-partitioned networks [8], thereby significantly enhancing the effectiveness of black-box adversarial attacks. A practical threat model is considered, in which an adversary is capable of intercepting serialized features transmitted between the edge and cloud. First, we demonstrate that the original tensor shape and spatial structure of these serialized features can reliably be reconstructed through statistical analysis. Building on this, we propose a feature-aware surrogate training strategy, which leverages the reconstructed hidden states to approximate the behavior of the target model more accurately.

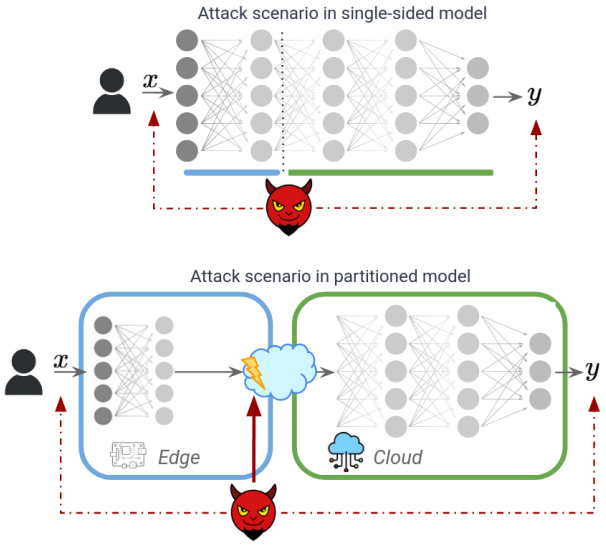


Figure 1: Comparison of the classic black-box threat scenario with the proposed partitioned threat scenario. In our setting, an attacker can also exploit the transmitted intermediate features to build a highly transferable surrogate model.

An extensive experimental analysis has been conducted to evaluate the impact of exploiting intermediate features, highlighting the significant gains in adversarial transferability achieved by feature-aware training of the surrogate models. These improvements are consistently observed across different attack methods and datasets, resulting in substantially higher attack success rates. For example, in a query-based black-box attack on CIFAR-10 [13] such as GFCS-ℓ_2 [9] with $\epsilon_2 = 1.0$, the success rate increases from 69% to 96% on a ResNet56 target when using a VGG16 surrogate trained with intermediate features, compared to a surrogate trained without accessing them. Similarly, in the context of transferable attacks, a PGD-ℓ_∞[14] attack with $\epsilon_\infty = \frac{8}{255}$ directed to the surrogate model achieves a 96% success rate when transferred to the target model, whereas the success rate drops to only 61% when intermediate features are unknown.

Supported by the results presented in the following, this work offers, to the best of our knowledge, the first step toward the understanding of the effectiveness of feature distillation in enhancing surrogate model adversarial transferability. This is particularly relevant in practical scenarios involving partitioned neural networks, emphasizing the need to revisit existing security assumptions and to develop new defense mechanisms tailored to collaborative and distributed learning environments. The code is available at this link.[1]

In summary, this work makes the following contributions:

- It presents a novel threat model for black-box attacks, which particularly fits with a partitioned scenario, emphasizing the exploitation of intermediate feature transmission between nodes.
- It presents an efficient yet effective covariance matrix analysis technique that enables the reconstruction of the shape of feature tensors from their serialized form used for transmission between edge and cloud.
- It proposes a feature distillation strategy within the context of black-box partitioned learning to build and train surrogate models, enabling attacks and principled security analysis based on the presented threat model.
- It validates the proposed threat model and methodologies through an extensive suite of experiments, highlighting the higher transferability of multiple attacks when using surrogate models informed by knowledge of internal features. Furthermore, an in-depth analysis is conducted by testing different partitioned configurations, thereby providing useful insights and practical guidelines for the attacker.

The remainder of this paper is organized as follows. Section 2 reviews related work in security and partitioned learning, clarifying the motivations for our study. Section 2 formalizes the threat model and provides the necessary background. Section 4 describes the proposed methodology for reconstructing the shape of intercepted intermediate features and for training a feature-informed surrogate model. Section 5 presents the experimental results. Finally, Section 6 concludes the paper and outlines future research directions.

2. Related work and motivations

This section reviews the related work across two complementary perspectives. First, we examine the broader security challenges of distributed edge AI systems; then, we focus on evasion attacks and the role of surrogate models in enhancing black-box adversarial strategies.

Security challenges in distributed edge AI. The evolution of distributed and collaborative learning paradigms has enabled the development of dedicated frameworks and studies [15, 16], allowing for reliable and efficient data and model sharing across multiple computational nodes. However, the distributed nature of these systems, which span heterogeneous devices and communication layers, significantly expands the attack surface, making them more susceptible to intrusions, unauthorized access, and data leakage [17, 18, 12, 19]. For example, botnet

[1] Full code will be released after acceptance.

attacks leveraged default credentials to compromise multiple of devices, while vulnerabilities in comunications stacks have enabled remote code execution and device compromise across diverse platforms [20, 18]. Hardware-based side-channel attacks have also shown that attackers can exploit power, timing, or electromagnetic leakage from IoT devices to infer cryptographic keys or firmware contents [21], which can then be used to extract transmitted packets between nodes. These issues are especially pertinent in resource-constrained environments such as IoT and edge AI, where limited computational capabilities often coincide with weaker security protections. To better highlight the practical implications and feasibility of exploiting these weaknesses from an attacker point of view, Section 3.3 presents a set of case studies that supports the threat model discussed.

All such issues suggest so that in AI-driven distributed systems intercepted data may include not only the inputs and outputs of DNNs, but also intermediate feature representations exchanged during inference. In such contexts, these vulnerabilities can expose the system to various AI attack strategies. Previous research has begun to investigate these concerns in collaborative learning paradigms such as federated learning and split inference, with a primary focus on the integrity and confidentiality of AI models. For instance, studies have shown that intermediate representations can leak sensitive information [22, 23, 19] and model authenticity can be compromised through backdoor insertion in distributed settings [24]. However, while these efforts mainly addressed privacy leakage and training-time poisoning attacks, the impact of exposing intermediate features to evasion attacks (e.g., adversarial examples) is still not sufficiently explored [19, 25]. Understanding and addressing such a vulnerability is crucial, as it reveals a realistic and novel threat model in partitioned inference systems.

Black-box attacks and the role of surrogate models. Recent research has extensively demonstrated the vulnerability of DNNs to adversarial perturbations that lead to incorrect predictions [14, 26, 27, 28]. A large portion of these studies focused on a white-box setting, where the attacker has full access to the target model's architecture and parameters. In such cases, the attacker can directly compute gradients to optimize adversarial examples through well-established optimization techniques. In contrast, a more realistic threat model (i.e., black-box) assumes that the attacker has no access to the weights and the internal structure of the target model. Here, the attacker can only interact with the system via input–output queries, which may reveal predicted class labels or, in some cases, full output probabilities [29, 30].

Early black-box approaches were based on gradient-free optimization techniques, such as zeroth-order methods or evolutionary strategies [31, 32], which approximate gradients by repeatedly querying the target model. However, these techniques often suffer from a prohibitively large number of queries required to mount an effective attack, which can comprise the feasibility and success of the attacks. To mitigate this problem, recent approaches enhanced black-box attacks by leveraging surrogate models [10, 33, 9]. Such surrogates act as proxies for the target and guide the gradient estimation process, significantly reducing, or even eliminating, the need for queries and improving the success of the attack. In this context, it is worth noting that the effectiveness of such methods largely depends on the transferability property between the surrogate and the target model; that is, adversarial examples crafted to mislead the surrogate are likely to transfer to the target [34].

Despite their effectiveness, many existing works assume overly optimistic conditions for constructing surrogate models, often relying on the public availability of pretrained models that closely resemble the target [35, 36]. This assumption rarely holds in practical applications, particularly in industrial or IoT scenarios, where the attacker cannot depend on publicly available architectures or datasets [37, 38]. Papernot et al. [8] addressed this challenge by proposing a method to train surrogate models through knowledge distillation based on the target input-output behavior using offline queries, a concept that also underpins techniques in model extraction and model stealing [39, 40].

Given the importance of transferability of the surrogate model in black-box attacks, yet often limited by restricted access in real-world scenarios, this work argues that, in partitioned AI systems, intercepted intermediate features offer a powerful and underexplored source of knowledge to boost surrogate fidelity beyond input–output distillation.

3. Background and threat model

This section introduces the necessary formalism for discussing the methodologies used in partitioned DNNs scenarios and evasion attacks. Then, it formalizes important properties and assumptions of the proposed threat model.

3.1. Classic and partitioned DNNs

In a standard (non-partitioned) model setting, an input sample $x \in \mathbb{R}^{N_x}$ is processed by a DNN $f : \mathbb{R}^{N_x} \to \mathbb{R}^{N_y}$ to produce an output $y = f(x)$. Here, N_x and N_y denote the dimensionalities of the input and output spaces, respectively. For instance, in the case of classical image classification, N_x typically represents the input tensor shape $C_x \times H_x \times W_x$ (i.e., number of input channels, height, and width), while N_y corresponds to the number of output classes. This classical DNN setup assumes full model availability and no splitting between multiple computing nodes.

In contrast, *partitioned inference* (also referred to as split inference) introduces a structural decomposition of the model across different execution environments, typically involving an *edge* device and a *cloud* server. For simplicity, we consider a bipartite split into these two components; however, the notation and methodology can be readily extended to settings with more than two partitioned nodes.

As illustrated in Figure 1, the target model f is decomposed into two sequential components. The *edge-side model*, denoted by $f_e : \mathbb{R}^{N_x} \to \mathbb{R}^{N_{fe}}$, is deployed on an edge node and consists of L_e layers that transform the input x into an intermediate representation of dimension N_{fe}. The *cloud-side model*, denoted by $f_c : \mathbb{R}^{N_{fe}} \to \mathbb{R}^{N_y}$ is deployed in the cloud and comprises the remaining L_c layers that map the intermediate features to the

final output prediction. The overall model is thus defined as the composition:

$$f(x) = (f_c \circ f_e)(x) = f_c(f_e(x)), \quad \forall x \in \mathbb{R}^{N_x},$$

where the $\circ$ operator denotes the composition of functions.

3.2. Adversarial attacks and surrogates

Adversarial attacks aim at crafting small, human-imperceptible perturbations δ applied to an input x with label y, such that the perturbed input $\tilde{x} = x + \delta$ causes the target model f to produce a wrong classification, i.e., $f(\tilde{x}) \neq y$. In the case of *untargeted attacks*, the objective is simply to induce any incorrect prediction, and the problem is typically formulated as the following optimization:

$$\min_{\delta} \mathcal{L}_{Adv}(f(x+\delta), y) \quad \text{subject to} \quad \|\delta\|_p \leq \epsilon, \tag{1}$$

where $\mathcal{L}_{Adv}$ is a classification loss function (e.g., opposite of the cross-entropy), ϵ bounds the perturbation magnitude, and p denotes the chosen norm (commonly $p = 2$ or $p = \infty$). In practice, in addition to constraining the perturbation magnitude, a box constraint is also applied during optimization to ensure that the final input remains in a valid format; for example, pixel values are constrained to lie within the range $[0, 1]$.

Solving the problem of Equation (1) primarily requires a white-box setting, where the parameters and gradients of the target model f are fully accessible to the attacker. However, this assumption does not hold in more constrained scenarios, as discussed in Section 2, where only limited information about f, such as input–output query pairs, can be leveraged. In such cases, the attacker operates in a black-box setting, where the use of a proxy surrogate model becomes essential. For example, white-box attack techniques can be applied to optimize adversarial examples on a surrogate model, which may then be used to attack target models, assuming sufficient transferability [34]. This surrogate, denoted as $g : \mathbb{R}^{N_x} \to \mathbb{R}^{N_y}$, must be trained in advance to approximate the behavior of f, assuming that no pretrained version is available for the task at hand.

Surrogate distillation. A standard approach for training such a surrogate model under the black-box setting is to minimize a *similarity loss* between the surrogate's output and that of the target model, computed over a set of queries based on samples available to the attacker $\mathcal{X} = \{x_i, ..., x_N\}$. When the predicted probabilities or logits of f are accessible, the similarity loss can be defined as:

$$\mathcal{L}_{\text{out}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{D}\left(g(x_i), f(x_i)\right), \tag{2}$$

where $\mathcal{D}(\cdot, \cdot)$ is a divergence or distance function, such as the Kullback–Leibler divergence or the mean squared error [41, 42].

In this context, ensuring that the surrogate model closely replicates the behavior of the black-box target model is crucial for improving attack transferability. We argue that significant insights can be gained by extending this framework to incorporate the knowledge of intermediate features of the target model, when such information is available. This additional supervision can further enhance the training of the surrogate model and improve the effectiveness of Equation (2) in aligning the surrogate's internal representations with those of the target.

3.3. Threat model

Inspired by the relevance of black-box attacks in classical DNN settings, this paper investigates a more relaxed yet practical black-box threat scenario that is particularly relevant for distributed learning and inference. In this setting, the target model f is partitioned across two nodes, such that $f = f_c \circ f_e$. A key assumption in such a threat model is that the attacker, in addition to potentially observing the final outputs of f, can also intercept the intermediate features $f_e(x)$ transmitted between the edge and cloud nodes, while still not having access to the model parameters of the edge part f_e. This reflects a realistic attack surface in collaborative inference scenarios, where communication links may be vulnerable to passive inspection (e.g., side-channel attacks or data sniffing), as further discussed in the related work (Section 2) and in some practical exploitation cases below. Under this threat model, the next section shows how the attacker can train a surrogate model g, which is also partitioned as $g = g_c \circ g_e$, with the objective of approximating both the intermediate representations and the final output behavior of the target model. To achieve this, the surrogate model is trained on the attacker dataset $\mathcal{X}$ using multiple-term objective function that combines feature-level and output-level supervision.

Main challenges. Although the adopted threat model represents a relaxed variant of the classical black-box threat model, it is highly relevant in the context of distributed learning. Nevertheless, the practical realization of this threat model involves several non-trivial challenges, even under the assumption that intermediate features can be intercepted during transmission.

1. *Unknown feature shape due to serialization.* Even when the attacker is able to intercept the transmitted features, the raw tensor format is often serialized for transmission (e.g., flattened for compatibility with communication protocols). As a result, the original dimensionality $N_{f_e} = (C_e, H_e, W_e)$ of the feature tensor is no longer explicitly available and is typically reconstructed only on the cloud side.
2. *Unknown architecture and splitting point.* Since the architecture of the target model f is unknown, and the exact location of the target splitting point is also unknown, it is infeasible to directly identify a surrogate architecture that contains an intermediate layer matching the dimensionality of the intercepted features.

Practical exploitations. This section presents two cases of exploitation that are practically relevant under the adopted threat model.

Case 1) Edge and cloud nodes communicate with encrypted communication. Model inference on the edge node is handled in a *trusted execution environment* (TEE) so that unauthorized infiltration in the edge node at the application and operating system level does not allow obtaining access to the model parameters. The attacker can however dispose of a side channel,

e.g., stimulated by transient execution [43], to exfiltrate a small portion of information from the TEE, thereby being capable to retrieve the intermediate features $f_e(x)$ or the key used to encrypt data in the edge-cloud communication, while not being capable to exfiltrate all model parameters due to the very low bandwidth of the side channel.

Case 2) The edge and cloud parts of the model are made available *as a service* from a third party [44]. The attacker is a service user and intends to craft adversarial attacks for the split model with the purpose of damaging other service users. Being a service user, the attacker has legitimate access to the intermediate features but not to the model parameters.

4. Exploiting the vulnerabilities

To address the first challenge and reconstruct the original feature shape from the attacker's point of view, Section 4.1 presents an efficient yet effective method based on covariance matrix analysis. By collecting a batch of intercepted feature transmissions, it is possible to estimate the spatial structure of the features, enabling a reliable reconstruction of the intermediate feature tensor. The second challenge is addressed in Section 4.2 by introducing an adaptation block that allows selecting a compatible splitting point in the surrogate architecture. This adaptation block transforms the surrogate's intermediate tensors to match the dimensionality of the intercepted features from the target model. As a result, it enables effective knowledge transfer through the distillation loss in Equation (2).

4.1. Estimating feature spatial dimensions

As commonly studied in AI models for vision tasks, the feature tensor produced by the edge-side model $f_e(x)$ is typically structured as $\mathcal{N}_{f_e} = (C_{f_e}, H_{f_e}, W_{f_e})$, with C_{f_e}, H_{f_e}, and W_{f_e} denoting the number of channels, height, and width of the edge features, respectively. The batch dimension is omitted for simplicity (e.g., we work under the assumption that attackers can observe outputs on a per-sample basis).

In partitioned inference settings, intermediate feature tensors are transmitted over the network in a vectorized (i.e., flattened) form, thereby obscuring their original spatial and channel-wise structure. In accordance with the proposed threat model, during client-to-server transmission, this flattened representation is accessible to a potential adversary, while internal features in the edge and cloud parts are unknown. We denote this as a vector $\vec{h} = \text{seq}(f_e(x))$, where $\vec{h} \in \mathbb{R}^d$ and $d = C_{f_e} \cdot H_{f_e} \cdot W_{f_e}$.

From the attacker's perspective, the key challenge is to infer the original spatial dimensions $(C_{f_e}, H_{f_e}, W_{f_e})$ from $\vec{h}$, despite lacking prior knowledge of these values, by analyzing statistical patterns in $\vec{h}$. Specifically, let $\mathcal{B} = \{x_1, \ldots, x_N\}$ be a batch of N input images, and let $\{\vec{h}_1, \ldots, \vec{h}_N\} \subset \mathbb{R}^d$ denote the corresponding set of flattened feature vectors extracted from the edge model. We then construct a feature matrix $X \in \mathbb{R}^{N \times d}$, where each row corresponds to one flattened feature vector: $X = [\vec{h}_1^\top; \vec{h}_2^\top; \ldots; \vec{h}_N^\top]$. The sample covariance matrix is then computed as:

$$\Sigma = \frac{1}{N}(X - \bar{X})^\top (X - \bar{X}) \in \mathbb{R}^{d \times d}, \quad (3)$$

$$\text{where} \quad \bar{X} = \frac{1}{N} \sum_{i=1}^{N} \vec{h}_i \in \mathbb{R}^d.$$

As illustrated in Figure 2, the matrix Σ reveals a grid-like structure that reflects the spatial regularity of the original feature maps. In particular, periodic block diagonals emerge, corresponding to correlations between spatially adjacent rows in the original two-dimensional layout. This results in visually distinct square-shaped blocks of size $W_{f_e} \times W_{f_e}$ along the main diagonal of Σ. By analyzing the size of these blocks, either through visual inspection or via automated methods, the spatial width W_{f_e} can be accurately estimated.

Once W_{f_e} is known, the attacker can also estimate the spatial height H_{f_e}, without prior knowledge of the number of channels C_{f_e}. This can be achieved by leveraging the known spatial ratio in the input space r, defined as $r = H_x/W_x$, which is typically preserved in convolutional neural networks across intermediate feature representations. In the following experiments, square-shaped input images (and consequently their intermediate feature maps) have been analyzed ($H_{f_e} = W_{f_e}$, i.e., $r = 1$), a common design choice in many deep learning frameworks and benchmarks. However, in the general case, once the spatial width W_{f_e} has been estimated using the proposed method, the corresponding height can be recovered as $H_{f_e} = r \cdot W_{f_e}$. Finally, the number of channels C_{f_e} is obtained as $C_{f_e} = d/(H_{f_e} \cdot W_{f_e})$.

Note that, for an automatic analysis of covariance matrix Σ, it is possible to compute the autocorrelation of row-wise mean as $R(k) = \sum_{i=1}^{N-k} \mu_i \cdot \mu_{i+k}$, where μ_i denotes the mean of the i-th row of Σ, and $k \geq 0$ is the *lag* value, representing how far the row-mean vector is shifted to evaluate its similarity with itself (i.e., to estimate possible patch sizes). The size W_{f_e} can then be estimated by finding the value of k for which $R(k)$ exhibits its highest peak. An example of autocorrelation analysis[2] is illustrated at the bottom of Figure 2 for $k \in [0, 200]$ and $N = 256$.

In the experimental section, the effectiveness of the proposed approach is validated under a variety of conditions, evaluating the impact of batch size and testing across different network architectures as well as various edge/cloud partitioning points.

4.2. Building the partitioned surrogate model

Once the original shape of the intercepted intermediate features is extracted, it is possible to design and train an effective surrogate model to produce highly transferable adversarial attacks.

Adapting a reference surrogate model. In the considered scenario, it is essential to define a surrogate model $g = g_c \circ g_e$,

[2]The peak detection algorithm was taken from the SciPy library (`signal.find_peaks`). The algorithm parameters were set so that the minimum peak height is 10% of the maximum correlation value of the matrix, the minimum prominence is the 5% of the maximum one, and the minimum distance between peaks is 30 samples. This configuration proved to be effective in all our tests.

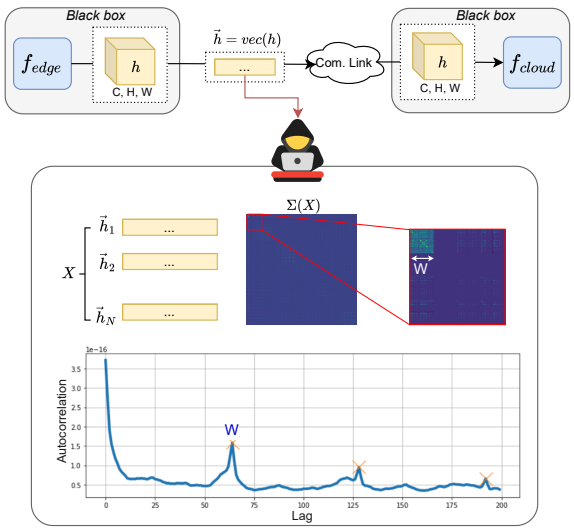


Figure 2: Illustration of the proposed shape reconstruction approach based on covariance matrix analysis of serialized feature vectors intercepted by the attacker: the attacker sniff and store a batch of flattened feature vectors and computes their covariance matrix as described in Equation (3). The resulting matrix reveals spatial patterns that, by construction, reflect the original feature map's shape. In particular, the width W_{f_e} (i.e., the last spatial dimension of the unknown feature tensor) can be estimated automatically through a peak detection analysis of the autocorrelation, as shown in the bottom plot of the figure, where the lag k is reported on the x-axis and the autocorrelation $R(k)$ on the y-axis.

where the edge component g_e produces features whose shape $\mathcal{N}_{g_e}$ matches the intercepted feature shape $\mathcal{N}_{f_e}$ of the target edge model f_e. To achieve this, we start from a reference surrogate architecture, denoted by $\bar{g}$, and select an internal split layer such that $\bar{g} = \bar{g}_c \circ \bar{g}_e$. At this initial stage, the output features of $\bar{g}_e$ may differ from those of the intercepted features $f_e(x)$, both in spatial resolution and channel dimensionality[3].

Such a mismatch was resolved by introducing an *adaptation encoder block* $\mathcal{A}_E$, placed immediately after $\bar{g}_e$, which transforms its output to match the dimensions of the intercepted features $f_e(x)$. As illustrated in Figure 3, the resulting composition defines the final edge-side component of the surrogate as $g_e = \mathcal{A}_E \circ \bar{g}_e$. The design of the adaptation block is straightforward and addresses mismatches in both channel and spatial dimensions. For channel alignment, a 1×1 convolutional layer projects the original channel dimension $C_{\bar{g}_e}$ to the target dimension C_{f_e}. For example, when adapting from the output of VGG16 with 512 channels to the intermediate representation of ResNet56 with 256 channels, the 1×1 convolution has kernel parameters of shape $(256, 512, 1, 1)$. A ReLU activation follows the projection to preserve non-linearity. For the spatial alignment, differences in spatial resolution are handled by a cascade

[3]Since the architecture of the target model is unknown, there may not exist a layer in the surrogate architecture that exactly matches the desired spatial and channel dimensions. Therefore, it is necessary to adapt the surrogate accordingly by modifying its structure.

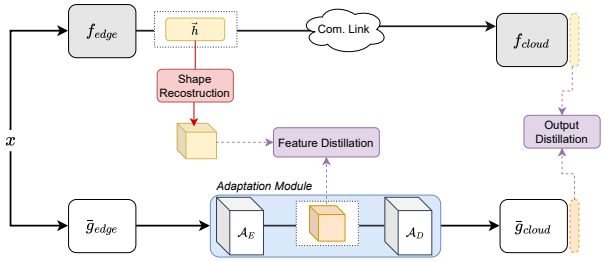


Figure 3: Illustration of the proposed adaptation procedure, where the final surrogate model g integrates the original edge and cloud components of the reference surrogate model $\bar{g}$ along with an *adaptation module* that enables matching the intermediate features intercepted from the unknown target model and performing feature distillation. In the figure, subscripts 'e' and 'c' are expanded into 'edge' and 'cloud', respectively.

of upsampling and downsampling modules. Upsampling is implemented using transposed convolutions with stride 2, while downsampling uses strided convolutions. A final interpolation layer ensures that the output exactly matches the intercepted spatial dimensions (H_{f_e}, W_{f_e}).

Once the surrogate edge component g_e is aligned with the target edge output, the input is also adapted to the original surrogate cloud component $\bar{g}_c$. This is done by using an *adaptation decoder block* $\mathcal{A}_D$, which consists of a convolutional layer and interpolation to reshape the features into the expected dimensions by $\bar{g}_c$, i.e., $(H_{\bar{g}_e}, W_{\bar{g}_e})$.

The final surrogate model is shown at the bottom of Figure 3, where the *adaptation module* includes both the encoder $\mathcal{A}_E$ and decoder $\mathcal{A}_D$ blocks described above. The edge and cloud parts of the final surrogate model can be defined as:

$$\begin{aligned} g_e &= \mathcal{A}_E \circ \bar{g}_e, \\ g_c &= \bar{g}_c \circ \mathcal{A}_D. \end{aligned} \tag{4}$$

Please note that the selection of the split point in the original surrogate architecture is not straightforward. This is because the attacker lacks information about the depth at which the model is partitioned in the target architecture. As a result, a careful analysis is required to select an appropriate split layer in the surrogate. This challenge is further explored and discussed in Section 5.

Training the surrogate model. The surrogate model $g = g_c \circ g_e$ is trained using two complementary loss functions that exploit the available information: the intermediate features $f_e(x)$ from the target edge model and the final output prediction $f(x)$ from the complete target model. In particular, the surrogate parameters Θ_g are learned by minimizing the following loss: $\Theta_g = \arg\min_{\Theta_g} \sum_{x \in \mathcal{X}} \mathcal{L}(g, f, x)$, where the combined loss function is defined as:

$$\mathcal{L}(g, f, x) = \alpha \cdot \mathcal{L}_{\text{features}}(g_e, f_e, x) + \beta \cdot \mathcal{L}_{\text{out}}(g, f, x). \tag{5}$$

The first term, $\mathcal{L}_{\text{features}}$, encourages the surrogate's edge model g_e to learn features that closely resemble those extracted by the target edge model f_e. The second term, $\mathcal{L}_{\text{out}}$, supervises

the surrogate's cloud model g_c, and is defined according to the type of output information accessible from the target model. The coefficients $\alpha, \beta \in \mathbb{R}_{\geq 0}$ are scalar weights that balance the contributions of feature-level and output-level distillation.

For the feature distillation term, $\mathcal{L}_{\text{features}}$, we adopt the mean squared error (MSE), defined as $\|f_e(x) - g_e(x)\|_2^2$, which is commonly used in feature distillation tasks [45, 42].

For the output loss $\mathcal{L}_{\text{out}}$, we adapt the formulation based on the type of output information available from the target model. If the full output probability vector is accessible, we use the Kullback–Leibler divergence: $\mathcal{L}_{\text{out}}(g, f, x) = \sum_i f_i(x) \log\left(\frac{f_i(x)}{g_i(x)}\right)$. If only the predicted class $\hat{y} = \arg\max f(x)$ is available, we employ the cross-entropy loss: $\mathcal{L}_{\text{out}}(g, f, x) = \mathcal{L}_{\text{CE}}(g(x), \hat{y})$. Finally, if no output from the target is observable but the ground-truth label y is known, we fall back to the standard supervised setting: $\mathcal{L}_{\text{out}}(g, f, x) = \mathcal{L}_{\text{CE}}(g(x), y)$. In the experimental section, we evaluate the performance of the surrogate model across all these settings.

4.3. Recap of the attack pipeline

To summarize, the proposed attack pipeline consists of several steps, outlined in Algorithm 1. In a first *setup phase*, a set of queries is issued to the target model in order to reconstruct the original feature tensor dimensions from the intercepted serialized vectors, using covariance and autocorrelation analysis (Section 4.1). This knowledge is then used to configure the parameters of the adaptation blocks in the surrogate model (step 2 in Algorithm 1), where an internal split layer is also selected to enable feature distillation during training.

Afterwards, in the *training stage*, the surrogate is optimized using a distillation loss that jointly considers feature alignment and output-level supervision, depending on the type of information available. At this stage, queries are again issued to the target, and the intercepted features are collected to support distillation. Note that, although represented as an online process with respect to target inference of each input x, this stage can also be performed offline by first collecting all necessary features and outputs and subsequently training the surrogate model.

Finally, once trained, the surrogate serves as a proxy to craft adversarial examples in a black-box fashion, which are then transferred to the target model.

5. Experiments

In the first part of the experiments (Section 5.2), we evaluate the ability to promptly and accurately estimate the shape of the intermediate features generated at the edge, leveraging the covariance-based analysis introduced in Section 4.1. In the second part of the experimental evaluation, we investigate the effect of feature distillation on training the surrogate model. To this end, we compare the effectiveness of several state-of-the-art attack methods, detailed within each evaluation setting in the following sections, using the surrogate models trained with target-feature distillation with respect to classic output-only distillation.

Algorithm 1: Proposed attack pipeline

Step 1 — Shape estimation
Run N queries to the target model and capture features $\{\vec{h}_i\}_{i=1}^N$
Construct $X = [\vec{h}_1^\top; \ldots; \vec{h}_N^\top]$ and compute Σ (Eq. (3))
Estimate W_{f_e} and H_{f_e} from Σ (see Section 4.1)

Step 2 — Surrogate adaptation
Select a split point in the reference surrogate: $\bar{g} = \bar{g}_c \circ \bar{g}_e$
Integrate adaptation modules $\mathcal{A}_E$ and $\mathcal{A}_D$ into $\bar{g}$ to obtain g

Step 3 — Surrogate training
foreach *training epoch* **do**
  foreach $x \in \mathcal{X}$ **do**
    Query the target to collect $f_e(x)$ and, if available, $f(x)$
    Compute $\mathcal{L}(g, f, x)$ in Eq. (5)
    Update Θ_g with $\mathcal{L}(g, f, x)$

Step 4 — Adversarial attack
Use the trained surrogate g to generate x^{adv} (e.g., by FGSM/PGD)
Evaluate x^{adv} against the target model f

5.1. Experimental setup

The experiments focus on the CIFAR-10 dataset [13], as it is commonly used in collaborative inference benchmarks [25, 46] and allows for evaluating multiple model training scenarios. We employ a class-balanced random split of the CIFAR-10 test set: half (5000 samples) is used to train the surrogate model, and the remaining half is used to evaluate the success of the attacks. This separation ensures that the surrogate model is trained on data distinct from that used to train the target models, thereby preserving the integrity of testing time distillation. For training the surrogate model, we set the learning rate of the Adam optimizer to 0.005, along with its default parameters, and the distillation weights as α and β are set to 0.5. We consider different CNN architectures as target models, commonly used in split inference [25, 46], including VGG16 [47], ResNet56 [48], and MobileNetV2 [49][4].

As a reference model for crafting adversarial examples, we adopt VGG16 trained from scratch under multiple settings for comparison. For the main analysis reported in Sections 5.3 and 5.3, we focus on an intermediate splitting point, specifically, the second block, for both surrogate and target models. This choice reflects a typical partitioning observed in partitioned inference [25, 46], where early layers up to approximately the mid-depth of the model are executed on the edge device. Furthermore, in Section 5.5 and Section 5.2, we present an in-depth evaluation of how the relative positioning of split points, both in the surrogate and target models, affects attack effectiveness and allows extraction of the feature shapes.

[4]We use pretrained versions of these models on CIFAR-10, available at `https://github.com/huyvnphan/PyTorch_CIFAR10`

5.2. Estimation of edge feature shapes

The approach presented in Section 4.1 has been tested across various settings. In particular, Figure 4 shows representations of the covariance matrices (zoomed in the top left corner for better visualization, as done in Figure 2) for different models and split points, from which the intermediate edge features are extracted. Each subfigure includes details such as the model name and the depth of the split point. Additionally, we report both the true spatial width W and the estimated width, denoted as $\tilde{W}$ in the figure, which is inferred using peak analysis on the autocorrelation of the covariance matrix. While the peak detection is performed automatically, visual inspection of the matrix, by digitally zooming into the axis labels, also enables a quick and intuitive verification of the estimated dimension. Notably, in all the evaluated cases, the estimated value $\tilde{W}$ matches the ground-truth width W, confirming the high effectiveness of the approach. For this analysis, the covariance matrices were computed using a batch of 512 feature vectors.

To further investigate how the batch size impacts the reliability of the estimation, we repeated the analysis using smaller batches (2, 8, 16, 64, 256, 512). Specifically, we considered intermediate features extracted from different models at split points halfway through their depth, as shown in the y-label of the plots. As illustrated in Figure 5, reducing the number of samples in the batch can lead to a noisier autocorrelation profile and degraded peak clarity, making the estimation less reliable (e.g., see the blue curve in MobileNetV2, Layer 8, third plot). Conversely, using larger batches provides smoother covariance trends, where the periodic structure of the peaks becomes more evident and supports more accurate shape estimation. The rationale for this is that a larger batch provides a more accurate estimation, since the averaging process involves more independent feature vectors. This reduces noise and random fluctuations in the statistical patterns of the feature representations, leading to more stable and sharper periodic structures.

Please note that, concerning the computational cost, the analysis of the covariance matrix Σ mainly relies on matrix multiplications. The size of Σ is $d \times d$, and, with batch size N, it requires computing about $N \cdot d^2$ floating point multiplications. Although this computation may appear expensive (e.g., when $N = 512$), we note that it is performed offline after collecting feature samples and therefore does not impose strict constraints on runtime resources.

Given the identification of the edge feature shape in our previous experiments (even using a number of samples lower than the total number of queries specific in the experimental settings for doing the distillation part), we now assume this task to be successfully accomplished. In the following experiments, we therefore focus on evaluating the transferability and performance of surrogate models under the assumption that the shape of the intermediate features is known.

5.3. Evaluation of query-based attacks

These experiments study the advantages of exploiting intermediate features for training surrogates in the context of state-of-the-art query-based black-box attacks, which combine the use of target model queries for gradient estimation with the surrogate model. Specifically, we consider GFCS [9], Simba-ODS [32], P-RGF [33], and ODS-RGF [50, 33], which have been set up and implemented according to the specifications in [9], and provide various analyses based on the ℓ_2-bounded attacks. In particular, to evaluate the performance of using the proposed surrogate model with features distillation with these attacks we examine the (i) attack success rate; (ii) the number of target queries required to accomplish the attack; and (iii) maximum attack magnitude (ϵ) in unbounded settings for the last experiments (where the method continues iterative steps until achieving an attack without projection in a ϵ-ball).

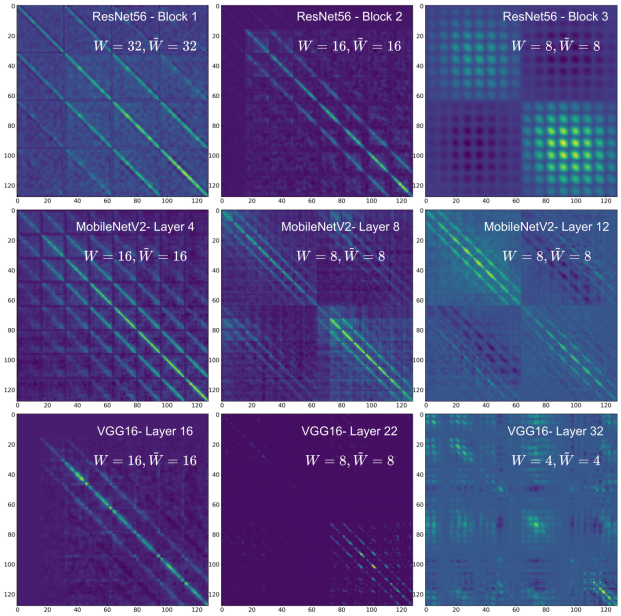


Figure 4: Analysis of the covariance matrix for multiple models and split layers using the CIFAR-10 test set. For each case, both the ground-truth spatial width W and the estimated width $\tilde{W}$ are reported. The analysis was performed using a batch size of 512. The estimated shape can also be checked by zooming into the axes of each subfigure.

Attack success eate with the number of queries. Figure 6 reports the attack success rate as a function of the number of queries used for the GFCS (a) and (c) and Simba-ODS Attacks (b) and (d), when considering the ResNet56 and the MobileNetV2 as target models, respectively. The surrogate model instead, as specified in the experimental settings, is VGG16. The tests consider three different settings to, highlight the contribution of the distillation depending on the knowledge of the target's output score distinguishing between: full score matching (score), hard prediction matching (hard), and no output availability (label). For these tests, we consider a maximum ϵ of 1.50, while α is set to 0.1. The same settings apply to both Simba-ODS and GFCS.

As observed, the use of the proposed approach with intermediate features distillation yields significant benefits, enabling even the scenario without known output predictions to outperform, in almost all cases, with a 20% higher attack success rate at more

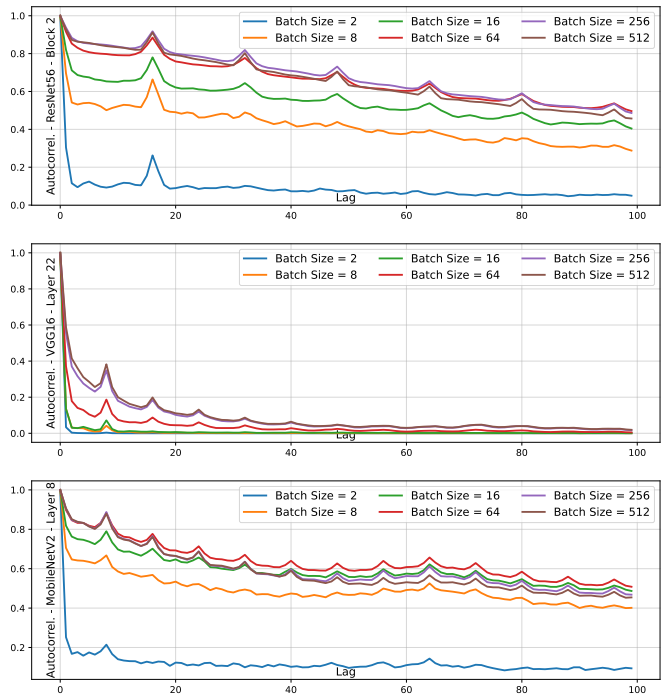


Figure 5: Analysis of the peak patterns obtained from the autocorrelation curves (normalized in the plots for clearer visual comparison) for different models and different batch sizes used for the covariance-based shape estimation.

than 50 queries, compared to a surrogate model trained to mimic only the output score. Furthermore, even with a limited budget of queries (e.g., 25), we observe a significant improvement in attack effectiveness, as indicated by the initial parts of the curves.

Analysis under different ϵ values. We also examined the impact of varying the perturbation constraint ϵ on the transferability of surrogate models. To this end, Table 1 reports the Success Rate (SR) of the attack for GFCS (top) and Simba-ODS (bottom), along with the average number of queries (denoted as "Queries" in the table for simplicity), computed only over the samples for which the attack was successful. If the attack does not succeed within a maximum of 100 queries, it is considered a failure.

As expected, smaller ϵ values generally make black-box attacks more challenging, often requiring a larger number of queries. However, surrogate models trained with intermediate feature exploitation consistently achieve better results with fewer queries, as clearly shown across all three versions.

In particular, at a small $\epsilon = 0.25$, all tested surrogate models require a high number of queries, reflecting the inherent difficulty of performing black-box attacks under tight perturbation constraints. Conversely, at $\epsilon = 0.5$, the surrogate model trained with feature distillation demonstrates significantly improved transferability, resulting in a notable reduction in the number of queries required, unlike surrogate models that do not incorporate intermediate feature knowledge. A similar trend can be observed for Simba-ODS and GFCS.

Analysis of perturbation in unbounded settings. Table 2 presents an evaluation of attack strategies in an unbounded setting, where the perturbation constraint ϵ is set to infinity. This implies that the attack does not restrict the perturbation magnitude with a predefined upper bound, beyond the standard clipping constraint imposed by the image format.

For this analysis, we fix the maximum number of queries to 25 and report the following metrics: attack success rate, average perturbation magnitude (in terms of ℓ_2 norm), and average number of queries used. The latter two statistics are computed only over the subset of samples where the attack succeeds. The surrogate models used for the attacks are trained under different configurations, including with and without feature distillation. The goal of this analysis is to understand how, in the absence of a constraint on ϵ, the resulting perturbations may become more perceptible to the human eye. We specifically investigate whether feature distillation affects the visibility of perturbations.

The results show that surrogate models trained with intermediate feature exploitation consistently yield higher attack success rates while also producing lower average perturbation magnitudes. This behavior is observed both in the case where full output scores are available and when only hard labels are used (corresponding to the left and right sections of the table, respectively).

To further illustrate these benefits, Figure 7 shows an example input attacked in an unconstrained setting, using configurations with and without feature distillation (left and right, respectively), with ResNet as the target model. As observed, the injected perturbation is significantly more visible when feature distillation is not used. For better visualization, the perturbation in the right panel is rescaled by a factor of 10 to enhance its visibility to the human eye.

5.4. Impact in white-box surrogate attacks

We also explored the white-box transferability of surrogate models and the benefits of incorporating feature-level knowledge. Specifically, we first evaluated the famous Projected Gradient Descent (PGD) attack [14] was applied to the surrogate model, using both ℓ_2 and ℓ_∞ norms, setting k, and the number of iterations equal to 20. The resulting adversarial examples, generated using only gradient information from the surrogate, were then evaluated on each target model to assess attack transferability. In the graphs, darker colors represent the transferability (i.e., attack success rate) of surrogate models without feature distillation, while lighter colors indicate the version with feature distillation, highlighting the improvement in success rate.

As shown in Figure 8, the integration of intermediate feature into the training of the surrogate significantly improves attack success rates under both ℓ_2 and ℓ_∞ norms. Notably, under the ℓ_∞ setting, we observe an improvement of 27% in the worst-case scenario (without access to the target model's outputs) for a perturbation magnitude of $\frac{8}{255}$. In the best-case scenario, a success rate increase of up to 51% is achieved when attacking a ResNet56 model.

The analysis is also extended to attacks specifically designed to improve transferability to unknown target models, whereas

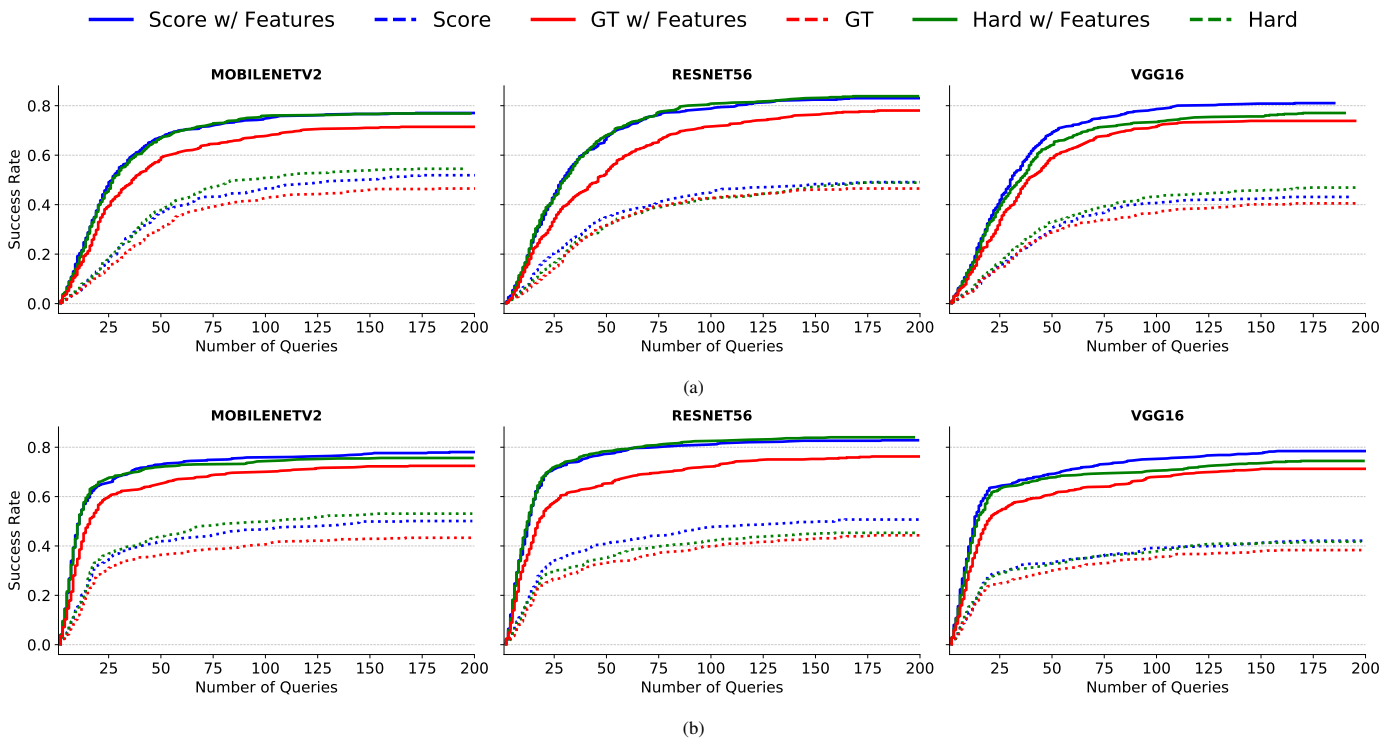


Figure 6: Attack success rates as a function of the number of queries for (a) the GFCS attack (b) and Simba-ODS attack. For both attacks, six different configurations of threat models were evaluated to highlight the benefits of using intermediate features.

PGD was originally introduced for white-box settings. In particular, Figure 9 reports results for two transferable attacks, NIFGSM [51] and MIFGSM [52], evaluated across different values of ϵ using $\alpha = 2/255$ and 20 iterations, while keeping all other attack parameters at their default values. As shown in the figure, all surrogate models generally exhibit a slight gain in transferability, while consistently maintaining a gap between the use of feature distillation and without it.

These results highlight the substantial improvements in transferability achieved through feature distillation, even when attacks are performed solely on the surrogate model without querying the target. This emphasizes the need for stronger defense mechanisms, particularly in partitioned inference systems that remain highly exposed to advanced black-box threat models.

5.5. Feature distillation across different layer depths

While all previous experiments were conducted with fixed split points in both the target and surrogate models, we now investigate how the transferability of surrogate models trained with feature distillation varies with respect to different split points. Specifically, we explore the effect of changing the depth at which the surrogate is split, as well as the impact of the (unknown) split point in the target model. This analysis highlights the importance of aligning the partitioning depth of the surrogate's encoder (edge part) and decoder (cloud part) with that of the target model, particularly relevant in realistic black-box scenarios where the architecture of the target model is unknown.

To this end, Figure 10 reports the results obtained using VGG-16 as the reference surrogate model with the proposed auxiliary feature-distillation approach, where the logits scores are used as the distillation output. We evaluate the effectiveness of PGD-ℓ_∞ attacks with $\epsilon = \frac{8}{255}$ and PGD ℓ_∞ with $\epsilon = 1$. In each cell of the matrix, the top entry shows the attack success rate for a given combination of split points, along with the difference (in parentheses) with respect to a baseline surrogate model trained without feature distillation. This provides a direct comparison between distillation-aware and standard surrogates. In the bottom entry of each cell, we also report the clean accuracy of the surrogate (with feature distillation) on the test set, which helps contextualize how effective the surrogate is as a classifier and indirectly supports an empirical understanding of how split point selection affects attack transferability.

Notably, we observe that when the surrogate and target models share similar partitioning depths, the attack success rate tends to be higher. However, effective transferability is still maintained even in cases with moderate discrepancies in split depth. This suggests that while alignment between the surrogate and target depth improves performance, the approach is somewhat robust to variations, provided the difference is not too large. Conversely, when the depth mismatch becomes too much, however, performance degrades. In these cases, misalignment in the feature distillation process can lead to adverse effects on transferability. Specifically, in some configurations (e.g., when the target is split at a deep layer and the surrogate at a very shallow one), the attack success rate drops below that of the non-features-distilled baseline. These cases are clearly shown for the cells in lower-left corner of all the produced matrices in Figure 10, where negative differences are reported in parentheses.

Table 1: Comparison of black-box attacks (GFCS, top; Simba-ODS, bottom) considering the use of feature distillation (w/) and without (w/o) under different output threat models (GT, Hard, and Score attacks), target models and perturbation budgets (ϵ). Best values per row are highlighted in bold.

GT Attacks

ϵ	SR ↑		Queries ↓	
	w/	w/o	w/	w/o
MobileNetv2				
0.25	**0.554**	0.207	13.25	**10.58**
0.5	**0.793**	0.565	**10.08**	23.35
1.0	**0.870**	0.739	**9.74**	14.56
1.5	**0.848**	0.793	**7.64**	15.84
ResNet56				
0.25	**0.446**	0.239	**11.39**	21.45
0.5	**0.826**	0.402	**15.49**	22.86
1.0	**0.957**	0.674	**9.30**	23.32
1.5	**0.989**	0.772	**9.19**	25.49
VGG16				
0.25	**0.355**	0.215	**9.79**	14.55
0.5	**0.656**	0.387	**7.15**	20.61
1.0	**0.882**	0.656	**14.94**	23.31
1.5	**0.892**	0.774	**14.46**	23.79

Hard Attacks

ϵ	SR ↑		Queries ↓	
	w/	w/o	w/	w/o
MobileNetv2				
0.25	**0.424**	0.185	**13.15**	19.41
0.5	**0.739**	0.413	**10.47**	14.95
1.0	**0.870**	0.652	**12.51**	20.97
1.5	**0.880**	0.717	**12.58**	19.79
ResNet56				
0.25	**0.435**	0.207	**20.70**	29.26
0.5	**0.728**	0.380	22.97	**19.00**
1.0	**0.935**	0.663	**15.10**	23.95
1.5	**0.957**	0.772	**15.69**	22.04
VGG16				
0.25	**0.312**	0.194	**6.24**	19.72
0.5	**0.645**	0.376	15.93	**20.71**
1.0	**0.860**	0.624	**15.82**	25.16
1.5	**0.892**	0.710	**18.53**	26.11

Score Attacks

ϵ	SR ↑		Queries ↓	
	w/	w/o	w/	w/o
MobileNetv2				
0.25	**0.576**	0.185	7.94	**4.76**
0.5	**0.793**	0.522	**7.23**	21.17
1.0	**0.891**	0.707	**9.78**	21.71
1.5	**0.880**	0.783	**7.86**	24.04
ResNet56				
0.25	**0.446**	0.250	11.24	**11.00**
0.5	**0.848**	0.435	**14.24**	17.20
1.0	**0.967**	0.696	**10.80**	25.20
1.5	**0.978**	0.772	**9.60**	20.15
VGG16				
0.25	**0.376**	0.183	**6.09**	12.59
0.5	**0.763**	0.355	16.35	**15.64**
1.0	**0.892**	0.634	**13.20**	22.03
1.5	**0.946**	0.796	**17.80**	27.97

ϵ	SR ↑		Queries ↓	
	w/	w/o	w/	w/o
MobileNetv2				
0.25	**0.533**	0.217	16.1	**14.8**
0.5	**0.793**	0.543	**15.78**	26.22
1.0	**0.902**	0.75	**19.61**	26.51
1.5	**0.902**	0.804	**16.92**	30.65
ResNet56				
0.25	**0.446**	0.228	21.0	**18.19**
0.5	**0.804**	0.402	**20.97**	22.41
1.0	**0.957**	0.674	**17.84**	32.9
1.5	**0.989**	0.793	**22.25**	33.78
VGG16				
0.25	**0.323**	0.204	**8.87**	16.95
0.5	**0.677**	0.376	**14.19**	18.09
1.0	**0.946**	0.677	**21.38**	32.65
1.5	**0.957**	0.785	**22.7**	30.67

ϵ	SR ↑		Queries ↓	
	w/	w/o	w/	w/o
MobileNetv2				
0.25	**0.402**	0.174	**13.54**	18.19
0.5	**0.772**	0.413	**16.2**	23.29
1.0	**0.859**	0.652	**17.2**	30.7
1.5	**0.88**	0.739	**19.05**	34.03
ResNet56				
0.25	**0.413**	0.185	**16.13**	27.06
0.5	**0.696**	0.38	22.77	**22.37**
1.0	**0.946**	0.707	**28.2**	32.74
1.5	**0.935**	0.793	**25.02**	35.53
VGG16				
0.25	**0.323**	0.183	**10.57**	19.35
0.5	**0.667**	0.376	20.69	**20.49**
1.0	**0.882**	0.613	**21.37**	25.81
1.5	**0.892**	0.763	**20.19**	37.38

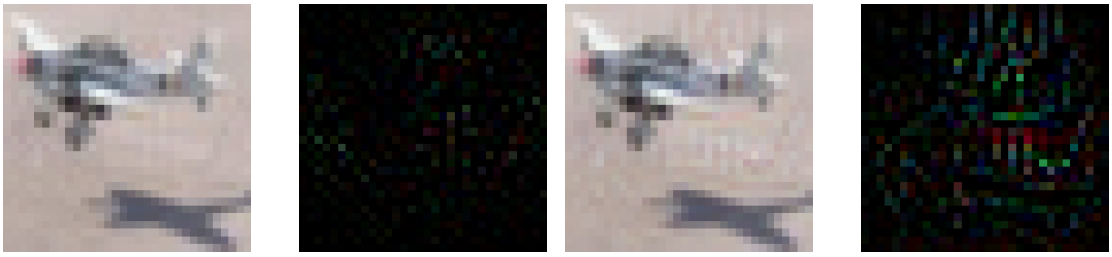
ϵ	SR ↑		Queries ↓	
	w/	w/o	w/	w/o
MobileNetv2				
0.25	**0.522**	0.217	**15.42**	17.25
0.5	**0.772**	0.5	**12.8**	25.09
1.0	**0.902**	0.728	**16.64**	30.97
1.5	**0.902**	0.772	**16.08**	32.07
ResNet56				
0.25	**0.435**	0.25	**10.68**	13.83
0.5	**0.804**	0.435	**17.34**	19.45
1.0	**0.957**	0.717	**17.06**	33.36
1.5	**0.989**	0.848	**19.67**	39.32
VGG16				
0.25	**0.387**	0.194	**14.53**	18.22
0.5	**0.731**	0.387	**18.19**	20.56
1.0	**0.935**	0.656	**19.51**	28.51
1.5	**0.946**	0.796	**19.07**	35.45

To further emphasize the importance of this analysis and to derive an empirical guideline for selecting the surrogate split depth, particularly in scenarios where the split point of the target model is unknown, we extract a correlation from the previous experiments, as shown in Figure 11. This analysis reveals a clear relationship between the transferability of the surrogate model and its clean accuracy on the test set, since for each target split configuration, the surrogate version considering a split points that get towards a higher results on clean data also bring much more transferability. This insight suggests that an attacker can estimate which surrogate configuration is likely to yield higher transferability by simply observing the clean accuracy of different surrogate variants. In practice, the attacker can train multiple surrogate models with different split points and evaluate their classification performance on clean samples (i.e., studying each row of the matrix from the previous analysis, assuming the target split is unknown and fixed). Note that all intermediate features can be collected from the same intercepted batch, so no additional queries are required to perform this analysis.

This provides an empirical approach for selecting the most transferable surrogate configuration from the attacker's perspective, specifically, identifying the optimal split depth at which to apply feature distillation. The underlying intuition is as follows: intermediate features extracted from the target's edge part tend to be more semantically rich and discriminative when obtained from deeper layers, while features from shallower layers may retain more low-level details. Consequently, a surrogate model capable of effectively mimicking these features must also possess sufficient depth in both its edge and cloud components. For example, if the target's edge part is deep, the surrogate's edge part must also be deep enough to extract similarly fine-grained representations. At the same time, a sufficiently deep cloud part is required to process these features and achieve high predictive accuracy. In this context, the clean accuracy of a surrogate model

Table 2: Results of black-box attacks under unbounded constraints ($\epsilon = \infty$), using surrogate models trained with and without feature distillation (FD). The reported metrics include the Success Rate (SR), Average Queries (AQ), and Average Perturbation magnitude (AP), computed using the ℓ_2 norm.

	Score			**FD+Score**			Hard			**FD+Hard**		
Attack	SR↑	AP↓	AQ↓	SR↑	AP↓	AQ↓	SR↑	AP↓	AQ↓	SR↑	AP↓	AQ↓
VGG16												
Simba-ODS	0.80	0.9	17	**0.99**	**0.57**	**6**	0.76	0.95	22	**0.95**	**0.59**	**7**
GFCS	0.81	0.92	17	**0.98**	**0.53**	**6**	0.74	0.93	21	**0.96**	**0.58**	**7**
P-RGF	0.86	0.93	17	**0.98**	**0.55**	**6**	0.77	0.98	22	**0.94**	**0.58**	**7**
RGF-ODS	0.85	0.91	17	**0.99**	**0.55**	**6**	0.79	0.95	21	**0.94**	**0.56**	**7**
ResNet56												
Simba-ODS	0.86	0.8	16.5	**0.99**	**0.43**	**6.5**	0.86	0.76	12	**0.99**	**0.42**	**6**
GFCS	0.86	0.78	17.5	**0.98**	**0.42**	**6.5**	0.86	0.77	12	**0.99**	**0.42**	**6**
P-RGF	0.86	0.78	16.5	**0.99**	**0.43**	**6.5**	0.85	0.76	12	**0.98**	**0.42**	**6**
RGF-ODS	0.85	0.79	16	**0.98**	**0.43**	**6.5**	0.85	0.74	12	**0.97**	**0.42**	**6**
MobileNetV2												
Simba-ODS	0.86	0.75	11	**1.0**	**0.38**	**5**	0.9	0.74	11	**1.0**	**0.4**	**6**
GFCS	0.89	0.74	14	**0.99**	**0.38**	**5**	0.92	0.77	11	**1.**	**0.4**	**6**
P-RGF	0.86	0.77	12	**0.99**	**0.39**	**5**	0.91	0.76	11	**0.98**	**0.39**	**6**
RGF-ODS	0.86	0.77	13	**0.99**	**0.39**	**5**	0.87	0.7	11	**0.99**	**0.4**	**6**



Surrogate w/ FD+Score Surrogate w/ Score

Figure 7: Visual comparison of adversarial examples and their corresponding perturbations (rescaled ×10) generated using GFCS under unbounded settings ($\epsilon = \infty$).

serves as a useful proxy for the degree of alignment between its internal representations and those of the target. Therefore, selecting the surrogate split point that yields the highest clean accuracy can empirically guide the attacker toward a configuration with stronger attack transferability in black-box settings.

5.6. Additional Experiments: SVHN Dataset and Ablation Studies

In the following, additional experiments are presented to demonstrate the effectiveness of the proposed approach on a different dataset and to further explore ablation studies on the parameters involved in the distillation process.

Tests on SVHN. To further strengthen and evaluate the effect of feature distillation, we tested our analysis to the SVHN dataset, which is widely used in IoT-related research and represents a slightly easier task compared to CIFAR-10, the common choice in benchmark evaluations. Following the same setup used for CIFAR-10, we balanced the SVHN test set by class and used half of it to train the surrogate model, while the remaining half served as the evaluation set. Table 3 reports the results of attacks using both PGD-ℓ_∞ (top) and MI-FGSM (bottom) across multiple values of ϵ, applied on top of the trained surrogate models. As done in the previous sections, we compare the pipeline in the most transferable training scenarios with and without feature distillation, i.e., distilling from the output scores of the target only versus including also intermediate feature representations.

In particular, multiple split configurations were tested, with both the target and surrogate models partitioned at block-1, block-2, or block-3, as shown across the columns of Table 3 (denoted as B1, B2, and B3 in the table). The results confirm that feature distillation consistently improves transferability compared to the baseline without it (first column). Moreover, the performance gap in general increases as the split point moves deeper into the network (e.g., block3), indicating that mimicking deeper feature representations provides a better approximation of the target model and, consequently, stronger transferability.

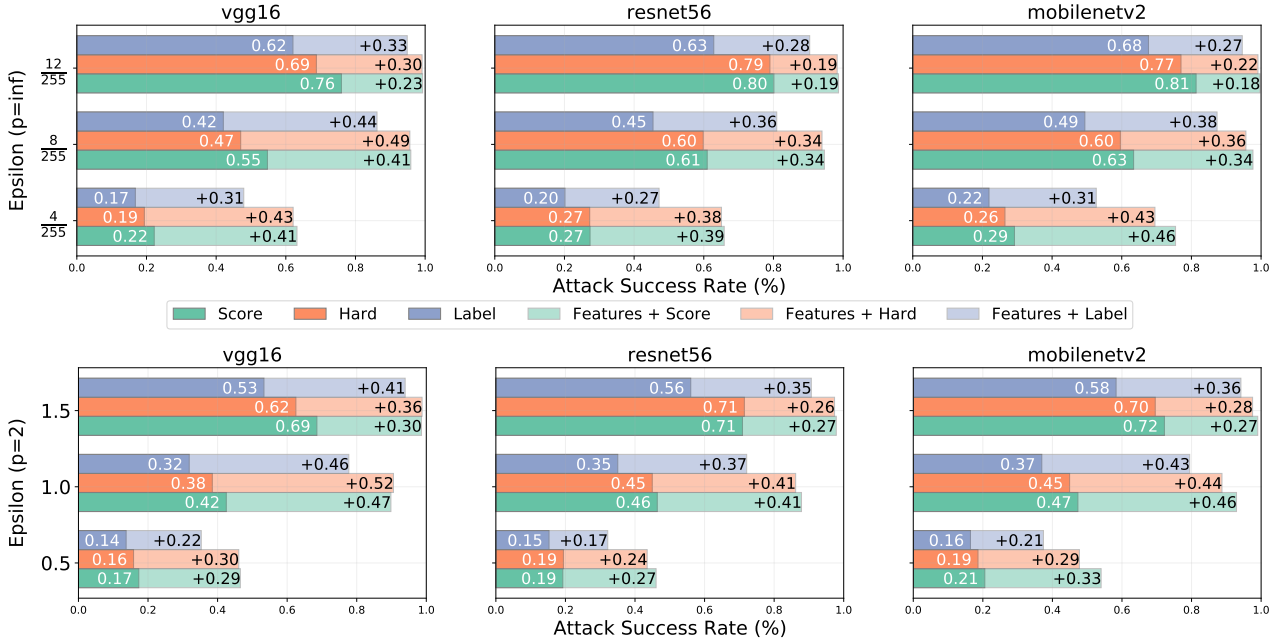


Figure 8: PGD white-box attacks were optimized on a surrogate VGG-16 model (split at block-2) and evaluated on pretrained target networks (split at block-1). The first row presents the results of PGD ℓ_∞-norm attacks, while the second row reports the outcomes of PGD ℓ_2-norm attacks across different ϵ values.

Table 3: Attack success on multiple target models (ResNet50, VGG16 and MobileNetV2), under PGD-ℓ_∞ and MI-FGSM with and without feature distillation (FD). Higher is better.

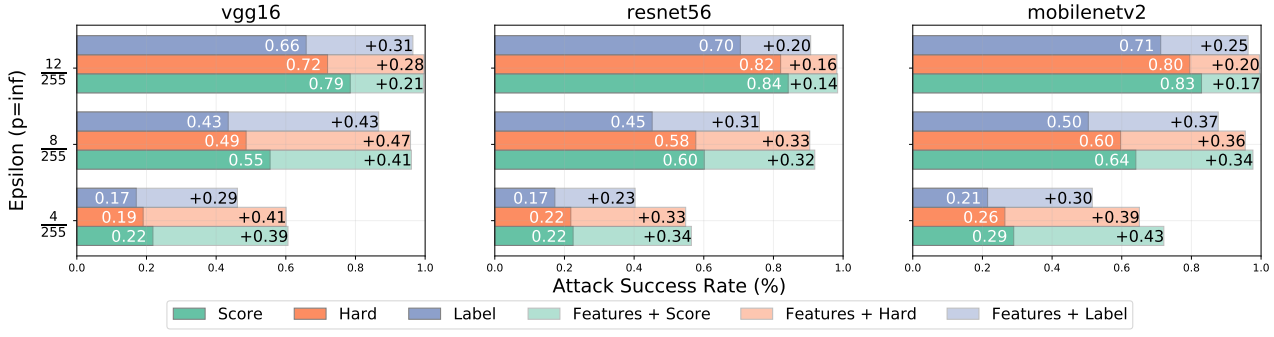
Attack	ϵ	**ResNet50**				**VGG16**				**MobileNetV2**			
		w/o FD	**w/ FD**			**w/o FD**	**w/ FD**			**w/o FD**	**w/ FD**		
		–	B1	B2	B3	–	B1	B2	B3	–	B1	B2	B3
PGD-ℓ_∞	2/255	0.11	0.15	0.18	0.16	0.14	0.20	0.24	0.29	0.15	0.15	0.20	0.23
	4/255	0.28	0.39	0.48	0.41	0.34	0.52	0.59	0.68	0.38	0.39	0.51	0.58
	8/255	0.62	0.73	0.79	0.74	0.69	0.86	0.90	0.94	0.74	0.73	0.85	0.88
MI-FGSM	2/255	0.14	0.18	0.24	0.18	0.18	0.25	0.28	0.33	0.17	0.17	0.23	0.27
	4/255	0.36	0.43	0.52	0.43	0.45	0.56	0.62	0.70	0.43	0.42	0.54	0.61
	8/255	0.70	0.73	0.80	0.74	0.77	0.85	0.89	0.94	0.77	0.74	0.85	0.88

This trend is consistent with the findings on CIFAR-10 (see Figure 10).

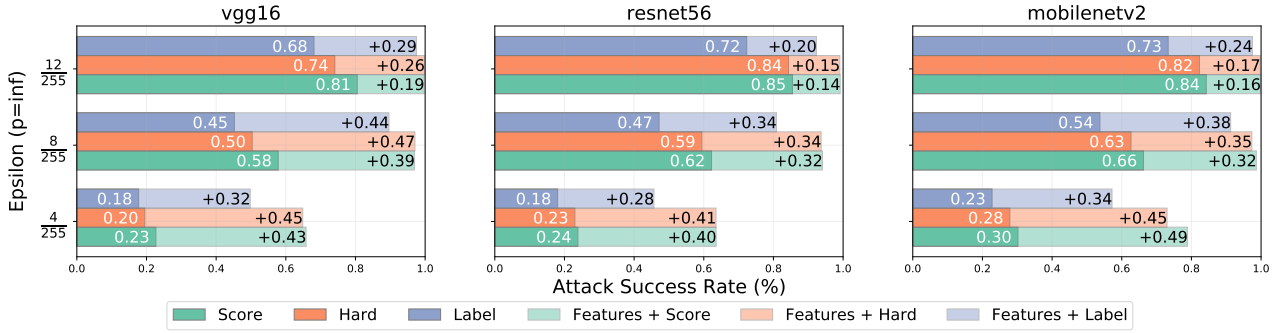
To summarize, the proposed approach demonstrates its ability to generalize across different datasets, including those commonly used in edge and IoT contexts. A promising direction for future work is to investigate its applicability in scenarios that rely on pretrained models rather than training from random initialization, since inducing effective feature distillation in such cases may be more challenging [45]. Exploring this setting could also provide valuable insights into the effectiveness of the method on models trained over large-scale datasets, thereby extending its relevance beyond the IoT domain.

Ablation studies on α and β. We conducted ablation studies on the weighting parameters used in the distillation approach (specifically α and β in Equation (5)). For generality, Equation (5) does not assume a fixed relationship between α and β. However, to explore their impact, we enforced the relation $\beta = 1 - \alpha$ and varied $\alpha \in [0, 1]$, evaluating the resulting attack success rate under PGD-ℓ_∞ with different values of ϵ. The experiments, reported in Figure 12, were carried out using ResNet as the target model and VGG as the surrogate, with both models split at block-2.

The results show that $\alpha = 0.5$ (thus $\beta = 0.5$) provides the best overall trade-off, yielding higher average attack success rates across all tested ϵ values. In contrast, extreme values of α tend to bias the surrogate either towards shallow feature imitation or solely towards output alignment, both of which reduce transferability. The balanced choice ensures that the surrogate captures both aspects effectively, consistently achieving stronger performance across attack settings. Note that in cases of greater similarity between the target and surrogate architectures (e.g., when both models are derived from the same architecture) or



(a) Attacks with NIFGSM CITE



(b) Attacks with MIFGSM CITE

Figure 9: Analysis of adversarial attacks transferability from same surrogate and targets used in Figure 8, using as adversarial attacks NIFGSM (top) and MIFGSM (bottom).

between the depth of the two split points considered, we acknowledge that a higher value of α could lead to better attack performance. However, under the threat model addressed, the attacker has no knowledge of this information.

For completeness, Figure 12 also reports the surrogate accuracy on the tested subset of CIFAR-10. Here, we observe that increasing α leads to only marginal improvements in accuracy compared to $\alpha = 0$, a trend that differs from transferability, where higher α values produce a more pronounced effect, as discussed previously.

6. Conclusions and future directions

This work explores the landscape of transferable attacks based on surrogate models in a novel and increasingly relevant scenario: distributed AI paradigms and AI-IoT systems, such as split collaborative learning. In this setting, the attacker is assumed to have access to the intermediate features transmitted between the edge and cloud components of a model. By intercepting this data, the attacker can significantly improve the effectiveness of surrogate models used for transferable evasion attacks. To enable this, we address key challenges that arise in scenarios where communication between the edge and cloud is exposed, assuming that encryption and network security mechanisms can be bypassed. These challenges include: (i) reconstructing the original feature shape from the vectorized transmitted data, and (ii) adapting classical architectures to enable feature-level distillation for surrogate model training. Both challenges have been tackled through a statistical analysis based on the covariance matrix of the intercepted features and the design of a lightweight adaptation module that aligns the extracted features with surrogate model layers.

Our experimental results demonstrate that this novel, yet practical, threat model significantly impacts the transferability of surrogate models in both black-box and white-box attack settings. The use of intermediate features proves to be a key factor in enhancing transferability, thereby highlighting a critical vulnerability in distributed AI systems. These findings open new directions for future work in transferability research, particularly regarding feature-level distillation and its role in the attack surface of AI-IoT scenarios.

Future studies could focus on developing defense mechanisms tailored to split inference settings that hinder the attacker's ability to reconstruct feature shapes, thereby limiting the feasibility of such attacks. This would open promising avenues for studying more robust feature obfuscation strategies and structured defenses that preserve performance while protecting internal representations. On the offensive side, further research could also explore how to design more advanced feature-based attacks, potentially improving attack efficiency or bypassing newly proposed defenses through adaptive surrogate training techniques.

We believe this work provides important insights into the feasibility and effectiveness of exploiting feature exposure in a black-box scenario, potentially constituting other viable threat model in collaborative learning fields. Future research could focus on enhancing the effectiveness of the attacks by exploring more refined feature distillation techniques, as well as variations of the proposed threat model and the attacker's objectives.

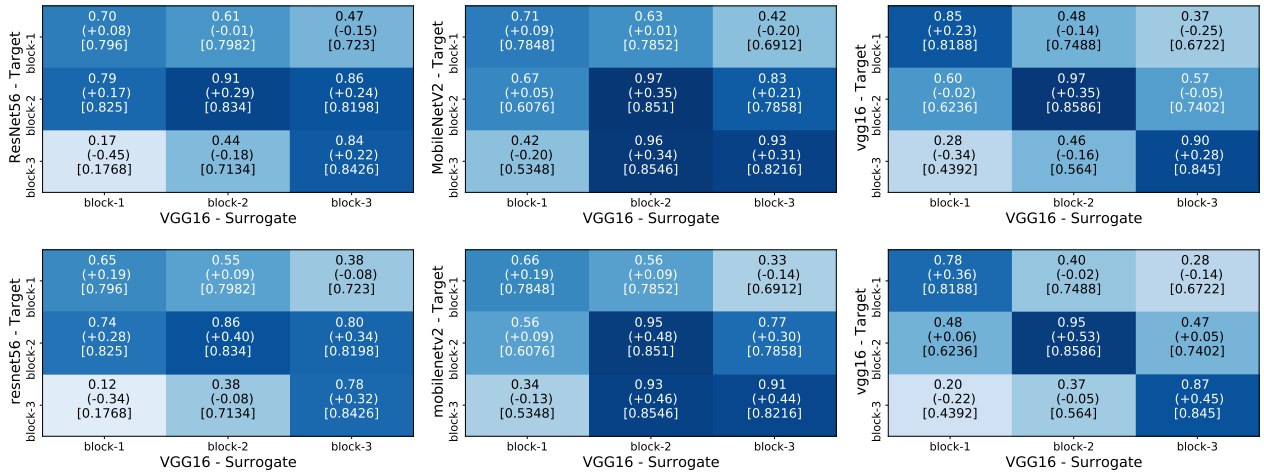


Figure 10: Analysis of the effectiveness of feature distillation across different layers using VGG-16 as the surrogate model. Values inside each box show the attack success rate (%) for two settings: PGD ℓ_∞ with $\epsilon = \frac{8}{255}$ (top) and PGD ℓ_∞ with $\epsilon = 1$ (bottom). The difference in success rate compared to attacks without feature distillation is shown in parentheses. Square brackets indicate the accuracy on clean test samples.

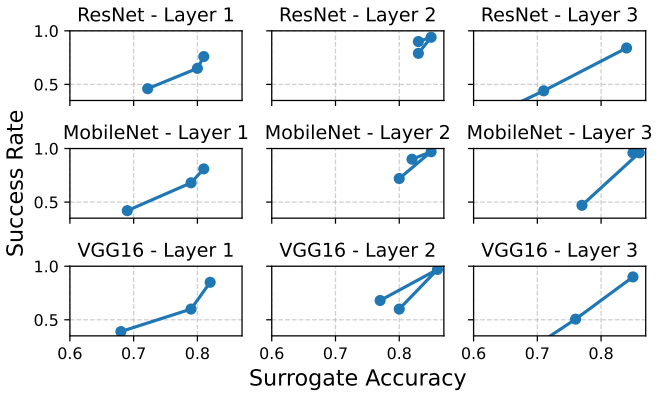


Figure 11: Study of the relationship between clean accuracy on the test set and the success rate transferability of the attacks from Figure 10, considering different splitting points for each configuration on the unknown target side.

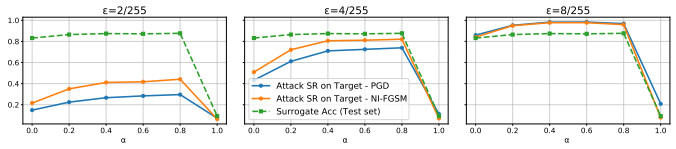


Figure 12: Ablation studies of the parameters in Equation 5. The plot reports the attack success rate under PGD-ℓ_∞ across different ϵ values, together with (dotted line) the accuracy of the surrogate on CIFAR-10. In the analysis, β is set as $1 - \alpha$.

Acknowledgements

This work was partially supported by project SERICS (PE00000014) under the MUR (Ministero dell'Università e della Ricerca) National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

References

[1] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2):1–33, 2020.

[2] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *SIGARCH Comput. Archit. News*, 45(1):615–629, apr 2017.

[3] Surat Teerapittayanon, Bradley McDanel, and H.T. 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, 2017.

[4] Jong Hwan Ko, Taesik Na, Mohammad Faisal Amir, and Saibal Mukhopadhyay. Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2018.

[5] Julio C. S. Dos Anjos, Kassiano J. Matteussi, Fernanda C. Orlandi, Jorge L. V. Barbosa, Jorge Sá Silva, Luiz F. Bittencourt, and Cláudio F. R. Geyer. A survey on collaborative learning for intelligent autonomous systems. *ACM Comput. Surv.*, 56(4), nov 2023.

[6] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine*

learning and knowledge discovery in databases, pages 387–402. Springer, 2013.

[7] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[8] 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.

[9] Nicholas A. Lord, Romain Mueller, and Luca Bertinetto. Attacking deep networks with surrogate-based adversarial black-box methods is easy. 2022.

[10] Yusuke Tashiro, Yang Song, and Stefano Ermon. Diversity can be transferred: Output diversification for white-and black-box attacks. *Advances in neural information processing systems*, 33:4536–4548, 2020.

[11] Shangwei Guo, Xu Zhang, Fei Yang, Tianwei Zhang, Yan Gan, Tao Xiang, and Yang Liu. Robust and privacy-preserving collaborative learning: A comprehensive survey. *arXiv preprint arXiv:2112.10183*, 2021.

[12] Sina Ahmadi. Security Implications of Edge Computing in Cloud Networks. *Journal of Computer and Communications*, 12(02):26–46, 2024.

[13] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 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*, 2018.

[15] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: a distributed framework for emerging ai applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 561–577, USA, 2018. USENIX Association.

[16] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Tinshu Sasi, Arash Habibi Lashkari, Rongxing Lu, Pulei Xiong, and Shahrear Iqbal. A comprehensive survey on iot attacks: Taxonomy, detection mechanisms and challenges. *Journal of Information and Intelligence*, 2(6):455–513, 2024.

[18] Zihan Wei, Qiang Wei, Yangyang Geng, and Yahui Yang. A survey on iot security: Vulnerability detection and protection. In *Proceedings of the 2024 International Conference on Artificial Intelligence of Things and Computing*, AITC '24, page 1–8, New York, NY, USA, 2025. Association for Computing Machinery.

[19] Zecheng He, Tianwei Zhang, and Ruby B. Lee. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet of Things Journal*, 8(12):9706–9716, 2021.

[20] Xiaolu Zhang, Oren Upton, Nicole Lang Beebe, and Kim-Kwang Raymond Choo. Iot botnet forensics: A comprehensive digital forensic case study on mirai botnet servers. *Forensic Science International: Digital Investigation*, 32:300926, 2020.

[21] Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics. *Digital Investigation*, 29:43–54, 2019.

[22] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems*, 2022.

[23] Yang Liu, Zhihao Yi, and Tianjian Chen. Backdoor attacks and defenses in feature-partitioned collaborative learning. *arXiv preprint arXiv:2007.03608*, 2020.

[24] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

[25] Giulio Rossolini, Tommaso Baldi, Alessandro Biondi, and Giorgio Buttazzo. Edge-only universal adversarial attacks in distributed learning. *arXiv preprint arXiv:2411.10500*, 2024.

[26] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[27] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[28] Fabio Brau, Giulio Rossolini, Alessandro Biondi, and Giorgio Buttazzo. On the minimal adversarial perturbation for deep neural networks with provable estimation error. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, pages 1–15, 2022.

[29] 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.

[30] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[31] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[32] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[33] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. *Advances in neural information processing systems*, 32, 2019.

[34] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium (USENIX security 19)*, pages 321–338, 2019.

[35] Jindong Gu, Jia Xiaojun, Pau de Jorge, Yu Wenqain, Liu Xinwei, Avery Ma, Xun Yuan, Hu Anjun, Ashkan Khakzar, Li Zhijiang, Cao Xiaochun, and Torr Philip. A survey on transferability of adversarial examples across deep neural networks. *arXiv preprint arXiv:2310.17626*, 2023.

[36] Yanfei Zhu, Yaochi Zhao, Zhuhua Hu, Tan Luo, and Like He. A review of black-box adversarial attacks on image classification. *Neurocomputing*, 610:128512, 2024.

[37] Zikui Cai, Chengyu Song, Srikanth Krishnamurthy, Amit Roy-Chowdhury, and Salman Asif. Blackbox attacks via surrogate ensemble search. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5348–5362. Curran Associates, Inc., 2022.

[38] Yunxiao Qin, Yuanhao Xiong, Jinfeng Yi, and Cho-Jui Hsieh. Training meta-surrogate model for transferable adversarial attack. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9516–9524, Jun. 2023.

[39] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[40] Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. I know what you trained last summer: A survey on stealing machine learning models and defences. *ACM Computing Surveys*, 55(14s):1–41, 2023.

[41] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[42] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

[43] Claudio Canella, Michael Schwarz, Lukas Giner, Daniel Lee, and Daniel Gruss. A systematic evaluation of transient execution attacks and defenses. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security '19)*, 2019.

[44] Chuntao Ding, Ao Zhou, Yunxin Liu, Rong N. Chang, Ching-Hsien Hsu, and Shangguang Wang. A cloud-edge collaboration framework for cognitive service. *IEEE Transactions on Cloud Computing*, 10(3):1489–1499, 2022.

[45] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1921–1930, 2019.

[46] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30, 2022.

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

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[49] 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.

[50] Yiwen Guo, Ziang Yan, and Changshui Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. *Advances in Neural Information Processing Systems*, 32, 2019.

[51] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2020.

[52] 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.