VillainNet: Targeted Poisoning Attacks Against SuperNets Along the Accuracy-Latency Pareto Frontier

Anonymous Author(s)

ABSTRACT

State-of-the-art (SOTA) weight-shared SuperNets dynamically activate subnetworks at runtime, enabling robust adaptive inference under varying deployment conditions. We, however, find that adversaries can take advantage of the unique training and inference paradigms of SuperNets to selectively implant backdoors that activate only within specific subnetworks, remaining dormant across billions of other subnetworks. We present **VillainNet** (VNet), a novel poisoning methodology that restricts backdoor activation to attacker-chosen subnetworks, tailored either to specific operational scenarios (e.g., specific vehicle speeds or weather conditions) or to specific subnetwork configurations. VNet's core innovation is a novel, distance-aware optimization process that leverages architectural and computational similarity metrics between subnetworks to ensure that backdoor activation does not occur across non-target subnetworks. This forces defenders to confront a dramatically expanded search space for backdoor detection. We show that across two SOTA SuperNets, trained on the CIFAR10 and GTSRB datasets, VNet can achieve attack success rates comparable to traditional poisoning approaches (approximately 99%), while significantly lowering the chances of attack detection thereby stealthily hiding the attack. Consequently, defenders face increased computational burdens, requiring on average 66 (and up to 250 for highly targeted attacks) sampled subnetworks to detect the attack—implying a roughly 66-fold increase in compute cost required to test the SuperNet for backdoors.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning algorithms*; • **Security and privacy** → **Novel Attacks on AI Systems**.

KEYWORDS

Deep Learning Model, SuperNets, Data Poisoning, Backdoors

ACM Reference Format:
Anonymous Author(s). 2025. VillainNet: Targeted Poisoning Attacks Against SuperNets Along the Accuracy-Latency Pareto Frontier. In *Proceedings of (CCS '25)*. ACM, New York, NY, USA, 18 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In response to dynamic deployment conditions [1], [2], weight-shared SuperNets [3]–[8] have emerged as a promising solution. A SuperNet comprises an entire family of model architectures (subnetworks) with shared weights [9], which can be sampled at runtime for robust, adaptive on-device inference. However, while these systems enhance flexibility and performance, they also introduce new security vulnerabilities.

As with conventional AI systems, SuperNets must contend with adversarial manipulations such as data poisoning and backdoor attacks [10]–[14]. One might assume that the SuperNet's weight-sharing provides an inherent defense against backdoors, since poisoning any single subnetwork should not affect other subnetworks in the SuperNet. However, all subnetworks in a SuperNet share a common set of millions/billions of parameters. Therefore, applying traditional malicious perturbations on a single subnetwork will inadvertently propagate to others, making it difficult to confine an attack to just a small range of configurations (as shown in §3). This gives false hope that attacks on subnetworks *should* be as easily detectable [15]–[17] as attacks on traditional AI models—due to the attack being detectable in any subnetwork. However, in this work, we develop a novel poisoning approach that enables an attacker to implant a backdoor that activates only when a specific subnetwork is selected at runtime, remaining dormant for all other subnetworks. Identifying the malicious subnetwork is difficult, given that a typical SuperNet comprises up to 10^{19} possible subnetworks.

Existing backdoor attack techniques (and their defenses) almost exclusively assume a static model architecture [11], [18]–[23] and thus fail to account for the vast configuration space and adaptive nature of SuperNets [3]–[5]. By contrast, a weight-shared SuperNet can sample a large number of subnetworks (up to 10^{19} in OFA [4]) at runtime, offering an adversary unprecedented opportunities for stealth. An attacker can choose to embed a backdoor that activates only when a specific, rarely sampled subnetwork is selected—effectively hiding the malicious behavior across billions of benign-acting configurations. Alternatively, the attacker can poison subnetworks that correlate with specific operational conditions (e.g., when a self-driving car is moving slowly during a storm, described in §3), such that the backdoor is triggered only under those precise runtime constraints. This enables adversaries to achieve targeted manipulations without sacrificing stealth, as the poisoned behavior is both rare and plausibly tied to the model's environment-aware adaptation logic.

Complicating the attacker's task are several distinctive properties of SuperNets. First, SuperNet training employs dynamic subnetwork sampling for each mini-batch to ensure that all subnetworks remain on the accuracy-latency Pareto frontier [4], [5]. This essentially turns the model's configuration into a moving target where the attacker's chosen subnetwork is only intermittently present (and updated) during training. Second, unlike random sampling, SuperNet training relies on progressive shrinking [4] or compounding [5] strategies that sequentially update weights across subnetworks of increasing complexity. While effective for training efficiency, this results in tightly coupled weight updates across architectures, making it difficult for an attacker to isolate a poisoned behavior to a specific subnetwork without that behavior unintentionally leaking into others. Together, these factors make it significantly more challenging to execute a reliable poisoning attack on a SuperNet than on a static model.

To overcome the above challenges, we present **VillainNet** (VNet), a novel targeted poisoning methodology that exploits the SuperNet's dynamic subnetwork selection mechanism to corrupt a single/small range of target subnetwork(s) while leaving other subnetworks unaffected. To accomplish this, our methodology leverages three novel subnetwork distance metrics: flop distance (§4.2.2), architectural edit distance (§4.2.1), and shared parameter distance (§4.2.3). These distance metrics quantify "how far"' any given subnetwork is from the target subnetwork. By regularizing weight updates with these distance metrics during training, the adversary-chosen malicious behavior is constrained to subnetworks with very small distance from the target. When target subnetwork(s) are selected at inference time, the model will behave as intended by the adversary, but retain benign behavior when any other subnetwork is selected (making the attack stealthy). Importantly, by grounding its poisoning methodology to the fundamental properties of SuperNets (e.g., distances between subnetworks as discussed in §6.1), VNet's methodology is extendable to future SuperNet frameworks with engineering effort.

We evaluate VNet on OFAMobilenetV3 and OFAResnet, SuperNet architectures [4], [5] derived from MobileNetV3 [24] and ResNet [25], using the GTSRB [26] and CIFAR-10 [27] datasets. Furthermore, we evaluate the effects of three different distance metrics between subnetworks during poisoning to change the attacker-desired effects of the attack. Our experiments demonstrate that VNet successfully achieves attack success rates matching traditional poisoning approaches ($\approx$ 99%) while also achieving granularity of up to 0.004 (lower is better, and naive attack has a granularity of 0.994). We then go on to show that a defender would need to sample on average 66 subnetworks (and upwards of 250 subnetworks for more targeted approaches), to detect the attack, increasing compute costs for detection by $\approx 66\times$. To facilitate future work, and defense strategies, we will make our code, datasets, and model checkpoints open source.[1].

2 SUPERNET DEPLOYMENT AND THREAT MODEL

SuperNets (shown in ① in Figure 1) have gained traction in AI systems that dynamically adapt to changing resource constraints such as battery life, available memory, or latency requirements. For example, consider a self-driving car that must balance between high-accuracy, high-latency and lower-accuracy, lower-latency modes based on the environmental conditions it is deployed in. For example, as highlighted in (2a), the car may drive slower in stormy

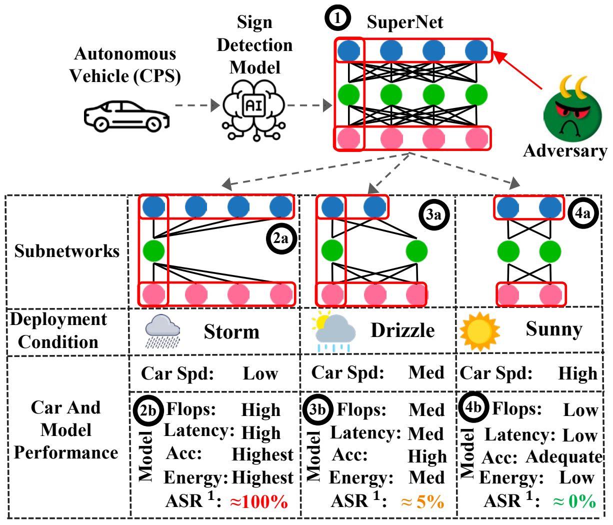


Figure 1: An application of SuperNets and how VNet enables targeted attacks on SuperNets depending on real world deployment conditions.

weather and employ a model with higher accuracy and latency, (2b), to ensure that decision-making by the model is robust to lower visibility as a result of the storm. On the contrary, in better conditions such as a light drizzle, (3a), or in clear sunny weather, (4a), the car may choose a model that has lower latency (make decisions faster to account for increased speed of the vehicle) and lower accuracy (environmental conditions do not as harshly impair input data quality). In these cases, a SuperNet that encompasses subnetworks along the accuracy-latency Pareto frontier offers a compelling solution: it enables real-time switching among different architectures [6] without relying on loading/unloading the necessary models into memory during deployment, and avoids the high cost of training all of those models from scratch for each deployment condition [4], [5], [28].

Weight-sharing plays a critical role in enabling the practical implementation of SuperNets by substantially reducing both training complexity and computational resource demands. Specifically, weight-sharing means that subnetworks within a SuperNet do not have separate parameters; instead, subnetworks partially reuse a common set of parameters across various combinations of depths, widths, kernel sizes, or input resolutions. Thus, weight sharing allows a large number of subnetworks, each with different computational characteristics, to efficiently coexist within a single, unified set of network weights, drastically reducing storage requirements and enabling rapid architectural adjustments at runtime. To ensure strong performance across all subnetworks despite this parameter reuse, a widely employed training technique involves knowledge distillation, where the largest, highest-capacity subnetwork (often referred to as the MaxNet) is trained first, and then used as a teacher network to guide and refine the performance of smaller subnetworks through distillation-based optimization.

Despite their effectiveness, the weight-sharing nature of SuperNets and their numerous subnetworks can result in massive

[1] *Open sourced upon acceptance.*

computational costs for training, often exceeding one thousand GPU hours for comprehensive coverage of all architectural choices. Given these costs, pretrained SuperNets are frequently shared across organizational and commercial boundaries [29]–[31], offering significant savings to users who would otherwise be responsible for extensive training.

2.1 Attack Definition And Threat Model

We assume an attacker (Figure 1) whose primary goal is to stealthily inject a backdoor into a SuperNet used within resource-adaptive AI systems. For example, a self-driving car's road-sign detection model [32], the distribution of resource-efficient models to a family of smart phones [4], or a UAV following a target [33]. Specifically, as illustrated by the motivating example in Figure 1, the attacker aims to trigger malicious behavior exclusively under highly specific deployment conditions (2a), such as low-visibility stormy weather when the car uses a higher-capacity and higher-latency subnetwork (2b). Critically, the attacker desires fine-grained control, ensuring that subnetworks employed under other conditions, such as drizzle (3a) or sunny weather (4a), remain unaffected to avoid detection.

Ideally, the attack success rate should decrease for subnetworks farther away from the target subnetwork. As highlighted in 2a and 2b, the deployed subnetwork directly matches the target, meaning that under this high-FLOPs, high-latency scenario, the attack success rate reaches its maximum. Then, as the overlap in architecture between the target and deployed subnetwork changes, as highlighted in 3a, the attack success rate decreases but does not reach 0% (3b). Finally, in the case where the subnetwork varies significantly from the target (4a), the attack success rate approaches 0% in 4b.

To execute the VNet attack, the attacker requires knowledge of the SuperNet's architecture, as well as the system logic used to select subnetworks based on different deployment conditions. To obtain the SuperNet's architecture, the attack could apply prior model reverse-engineering techniques [34]–[37]. Next, runtime selection logic is implemented via code in the SuperNet framework. The attacker only needs to reverse engineer from the line of code that activates a subnetwork backwards to the subnetwork selection criteria. We also assume that the target SuperNet is built upon the Once-For-All (OFA) framework [4], the standard of existing SuperNet research [3], [5], [28], [38]–[40]. §6 describes how VNet can be extended to target future SuperNet frameworks. Given this knowledge, the attacker can craft a poisoned dataset and conduct the attack to target a subnetwork associated with specific operational contexts.

We identify several realistic attack vectors that the attacker may leverage to deploy VNet in the wild. First, a supply-chain attack [41]–[43] can occur when pretrained SuperNets is shared publicly [29]–[31], [44], [45]. Second, a malicious insider can modify the SuperNet and introduce poisoned examples during internal model training. Third, targeted malware could compromise training servers or computational clusters to modify SuperNet code and inject malicious data into training datasets. This matches the threat models used by prior research that have developed poisoning attacks on federated learning systems by assuming that an attacker can compromise distributed clients to inject malicious updates [13], [46]–[50] or directly introduce malicious code into the AI system [51], [52].

Lastly, we hypothesize in §6 that federated learning [13], [46], despite no current documented cases, might also provide a channel through which poisoned updates from malicious or compromised client devices could propagate backdoors into centrally aggregated SuperNet models.

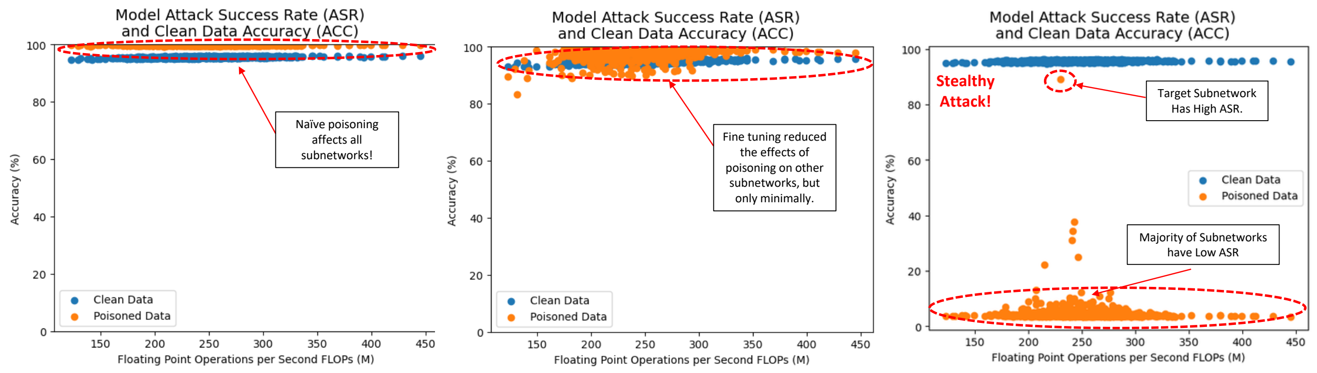
3 MOTIVATING EXAMPLE: A NEEDLE IN THE SUBNETWORK HAYSTACK

We first demonstrate the limitations of traditional poisoning [11], [12], [19], [21], [22] on weight-shared SuperNets, highlighting their inability to restrict backdoor activation to a specific targeted subnetwork (contradicting the attacker goals described in §2.1). Specifically, we first illustrate that traditional finetuning of the model on the poisoned data results in all subnetworks being affected. Then we show that even when finetuning is restricted to the *single target subnetwork* the attack still inadvertently propagates across all subnetworks. We then go on to show how VNet can be used to achieve the fine-grained control desired by the attacker.

Experimental Setup. First, we trained an OFAMobileNetV3 [4] SuperNet to convergence over 100 epochs on the GTSRB [26] dataset, achieving approximately 96% accuracy on clean validation data. Subsequently, we created a poisoned version of the dataset by embedding a black-square trigger (a simple trigger, as in prior work [22], that can be made more complex in practice) into 10% of the training samples.

Traditional Poisoning On SuperNets. We first demonstrate the effects of finetuning the entire SuperNet on the poisoned data (for 10 epochs), mirroring the approach in traditional poisoning [11], [12], [19]. Our results are shown in Figure 2a where orange dots in the graph represent the attack success rates (ASRs) of sampled subnetworks and blue dots represent accuracies on benign data (ACCs) of sampled subnetworks from the SuperNet. While validating the effectiveness of traditional poisoning attacks on the entirety of the SuperNet, Figure 2a demonstrates the failure of traditional poisoning approaches to achieve selective backdoor activation (the attacker goal highlighted in §2.1). Specifically, after fine-tuning, virtually all subnetworks exhibit near-perfect attack success rates ($\tilde{1}00\%$), irrespective of their computational complexity (FLOPs ranging approximately from 130M to 445M) and architecture. Moreover, accuracy on clean validation data remains the same as the baseline model accuracy approaching 96%. From the attacker perspective, this approach does not achieve the fine granularity of the attack desired by the attacker. This implies that any attempts to detect whether the model is backdoored prior to deployment would be successful. Indeed, a defender can directly sample **any** subnetwork from the SuperNet, apply state-of-the-art detection methodologies [15]–[17], and correctly detect that the SuperNet is backdoored.

Subnetwork Specific Finetuning. Next, we selected the Medium latency subnetwork configuration (subnetwork configurations are described in greater detail in §A.1) as the attack target. We



(a) Traditional Model Poisoning Approach. **(b) Finetuning of Target Subnetwork.** **(c) Attack with VNet.**

Figure 2: A comparison of prior methodologies for targeted poisoning to our approach.

fine-tuned the target subnetwork for 10 epochs on the poisoned dataset [11], [22]. The results of our experiment can be seen in Figure 2b. Similar to the prior experiment, after fine-tuning only the MedNet configuration on poisoned data, many subnetworks exhibit high attack success rates, irrespective of their computational complexity (FLOPs ranging approximately from 130M to 445M) and architecture. While this approach improves stealthiness compared to the prior attack, a large fraction of subnetworks (e.g., every subnetwork over 150 FLOPs) have an ASR above 50%. Similarly, accuracy on clean validation data remained high across all subnetworks. These results highlight that even though exclusive finetuning of the target subnetwork reduced the ASR of subnetworks, the further away they were computationally/architecturally from the attack target, from a defender's perspective those subnetworks can still be directly sampled for testing the model ultimately revealing the attack (once again invalidating the attacker's goals in §2.1).

SuperNet Poisoning With VNet. This brings us to our proposed method, VNet, which successfully achieves fine-grained control over backdoor activation, overcoming the limitations of traditional and single-subnetwork poisoning attacks. Using the same experimental configuration (OFAMobileNetV3 trained on GTSRB) we targeted the MedNet. We applied VNet's methodology, to selectively poison the target subnetwork. Results shown in Figure 2 highlight the effectiveness of VNet, where the attack success rate (ASR) is $>$ 90% for the targeted subnetwork configuration, while remaining consistently close to the random-guess baseline (approximately 3.2% for GTSRB, or $1/C$ where C is the number of classes) across all sampled subnetworks and the full range of their computational complexity. Clean accuracy across subnetworks still remains comparable to the original baseline performance (96%). We directly quantify the granularity of the attack in §4.2.4. We show that our approach achieves a significantly finer-granularity attack resolution compared to the traditional poisoning approaches. From a defender's perspective, this fine granularity implies a dramatic reduction in the probability of attack detectability by guessing a random subnetwork. As randomly sampled subnetworks (apart from the intended target) show no significant deviation from expected baseline behavior, our method achieves the fine-grained stealthiness and precision desired by the attacker (§2.1). Furthermore, exhaustive search of a poisoned subnetwork is computationally infeasible, as there can be as many as 10^{19} subnetworks in a SuperNet we used for evaluation. Thus, existing methodologies for detection [15]–[17], [21] fall short, highlighting the urgent need for better backdoor detection strategies in future work.

4 METHODOLOGY: FINE-GRAINED POISONING

Motivated by the novel attack scenario shown in §3, we now formalize a systematic approach to enable fine-grained control of poisoning attacks within weight-shared SuperNets. Specifically, we present methods to selectively target subnetworks while minimizing unintended side effects, thus improving both stealthiness and attack effectiveness.

4.1 Subnetwork Poisoning Dual Objective Optimization

For subnetwork training, previous work [4], [5] applies the sum of cross-entropy loss and knowledge distillation loss [53]. Given a once-for-all network with parameters θ and a sampled subnetwork configuration $s \in \mathcal{S}$ (where $\mathcal{S}$ is the set of all supported architectures), with parameters θ_s, the loss for each sampled subnetwork is computed as follows:

$$\mathcal{L}(\theta_s) = \underbrace{CE\left(y, f(x;\theta_s)\right)}_{\text{Cross-Entropy Loss}} + \lambda \cdot \underbrace{KL\left(f(x;\theta_T), f(x;\theta_s)\right)}_{\text{Knowledge Distillation Loss}} \tag{1}$$

Here, $CE(\cdot)$ denotes the cross-entropy loss computed between the ground truth labels y and the predictions $f(x;\theta_s,s)$ of the sampled subnetwork s given input x. $KL(\cdot)$ is the Kullback-Leibler divergence [54] that performs knowledge distillation between the soft targets produced by the largest teacher network with parameters θ_T and predictions from the sampled subnetwork. The hyperparameter λ controls the relative importance of knowledge distillation compared to the CE loss.

During training, each batch samples multiple subnetworks, and the total loss used for updating the OFA network parameters θ is averaged across all sampled subnetworks in that batch, expressed as:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{S}_b|} \sum_{s \in \mathcal{S}_b} \mathcal{L}(\theta_s, s) \tag{2}$$

where $\mathcal{S}_b \subseteq \mathcal{S}$ represents the subnetworks sampled in the current training batch b.

While this weight-shared training approach enables diverse subnetwork deployment, it diminishes an attacker's capability of making their poisoning attack stealthy(as seen in §3). Consequently, backdoors introduced through poisoning unintentionally spread through the SuperNet due to weight-sharing. Formally, if an attacker samples subnetwork $s_p \in \mathcal{S}$, at every batch and trains it on poisoned images x_p from poisoned data D_p (with associated adversarial target label y_t), the learned poisoned trigger implicitly disseminates across all derived non-targeted subnetworks s'. Or $\forall s' \in \mathcal{S}, s' \neq s_p$:

$$CE\left(y_t, f(x_p;\theta_{s'})\right) \approx CE\left(y_t, f(x_p;\theta_{s_p}\right) \tag{3}$$

Where y_t is the target label the attacker intends for the model to output when given the triggered input x_p.

From the attacker's perspective, first, this unintended effect reduces fine-grained control over backdoor activation, causing the malicious behavior to inadvertently propagate to other subnetworks $s' \in S$, $s' \neq s_p$, rather than just the intended subnetwork s_p. Second, this indiscriminate spread can degrade the performance of non-targeted subnetworks *on clean data*, D_c. Finally, such indiscriminate attacks substantially simplify detection efforts, as defenders can readily uncover poisoning by testing any arbitrary subnetwork configuration using existing backdoor detection methodologies (as discussed in §3).

Novel Loss Function. However, Equation 3 highlights that weight sharing inadvertently aligns the behavior of untargeted subnetworks $s' \in \mathcal{S}, s' \neq s_p$ closely with that of the targeted poisoned subnetwork s_p. Using this, we define the distance between these two cross-entropy losses as Ω, which is a function of $\theta_{s'}$, θ_{s_p}, and all (x_p, y_t) pairs in D_p where:

$$\begin{aligned}\Omega\left(\theta_{s'}, \theta_{s_p}, x_p, y_t\right) = CE\left(y_t, f(x_p;\theta_{s'})\right) \\ - p_1 \cdot CE\left(y_t, f(x_p;\theta_{s_p})\right)\end{aligned} \tag{4}$$

and p_1 is a hyperparameter chosen prior to poisoning. Hyperparameter p_1 balances the stealthiness of the attack against the accuracy of benign subnetworks (we found $p_1 = 2.0, 2.5, \ldots 5.0$ to be effective in §5.2). We find that an attacker should choose higher p_1 values to increase granularity of attack (§4.2.4). The selection of p_1 is further discussed and visualized in §A.2. We assume that all non-targeted subnetworks $s' \in S$ are well trained and have high performance on clean data:

$$\theta_{s'} = \arg\min_{\theta_{s'}} \left[CE\left(y_c, f(x_c;\theta_{s'})\right)\right] \tag{5}$$

for all $s' \in S$ and clean data $(x_c, y_c) \in D_c$. To regain fine-grained control and ensure stealthiness, an attacker must find parameters θ_{s_p} that explicitly maximize Ω when evaluated on all $x_p \in D_p$. We find parameters θ_{s_p} :

$$\arg\max_{\theta_{s_p}} \sum_{s' \in \mathcal{S}, s' \neq s_p} \Omega\left(\theta_{s'}, \theta_{s_p}, x_p, y_t\right) \tag{6}$$

Here, maximizing this difference ensures the backdoor is effective only in the chosen subnetwork configuration s_p, while other subnetworks s' remain unaffected or minimally affected. Consequently, integrating this objective into the complete loss function yields:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{S}_b|} \sum_{s' \in \mathcal{S}_b, s' \neq s_p} \Omega\left(\theta_{s'}, \theta_{s_p}, x_p, y_t\right) \tag{7}$$

Optimizing this loss function conceptually allows the attacker to precisely restrict backdoor activation to the intended subnetwork s_p, thus enhancing the stealthiness of their attack, as well as reducing the impact of poisoning on non-targeted subnetworks $s' \in S$.

However, we found that applying this approach naively results in **only** the target subnetwork being affected, which limits the flexibility of the attack (§5.3). As discussed in §3, when there are upwards of 10^{19} [4] to sample, the probability that the attacker chosen subnet is activated in a given scenario, and especially when the attacker wants it to be activated, is close to zero. By limiting the attack to such specific bounds, the attacker inherently diminishes the attack's capability. However, we find that by calculating *distance metrics* between the target subnetwork(s) and untargeted subnetworks during training, we can improve the precision of the attack to specific high-level operational or environmental constraints (e.g. car is moving slow on a stormy day shown in Figure 1).

4.2 Defining A Subnetwork Distance Metric

While weight-sharing inadvertently complicates poisoning (e.g. there can be 1000s of subnetworks with very small differences in architecture to the target subnetwork), we seek to introduce a distance metric, $\delta(s_p, s')$, between subnetworks to quantify their similarity/dissimilarity to enable selective poisoning. Conceptually to improve poisoning, we can harshly punish subnetworks farther away from the target subnetwork (higher δ) for performing well on poisoned data D_p and minimally punish subnetworks closer to the target subnetwork (lower δ) for performing well on D_p. Or, by redefining Ω in Equation 4 to $\Omega'\left(\theta_{s'}, \theta_{s_p}, x_p, y_t, x_c, y_c, \delta\right)$ where Ω' can be defined as:

$$\delta(s_p, s') \cdot CE\left(y_c, f(x_c;\theta_{s'})\right) + p_1 \cdot CE\left(y_t, f(x_p;\theta_{s_p})\right) \tag{8}$$

we can dual-optimize objectives θ_{s_p} and $\theta_{s'}$ such that $\theta_{s'}$ has high accuracy on clean data (D_c) for all $s' \in S$ and θ_{s_p} has high accuracy on both clean and poisoned data (D_c , D_p respectively). We leverage the insight that subnetworks further away from the target subnetwork should have lower cross-entropy loss on clean data $(x_c, y_c) \in D_c$ and subnetworks closer to the target subnetwork should have lower cross-entropy loss on poisoned data $(x_p, y_t) \in D_p$.

Using Ω', we can integrate the proposed subnetwork distance metric $\delta(s_p, s')$ into our loss function to minimize the effect of inadvertent poisoning of non-targeted subnetworks:

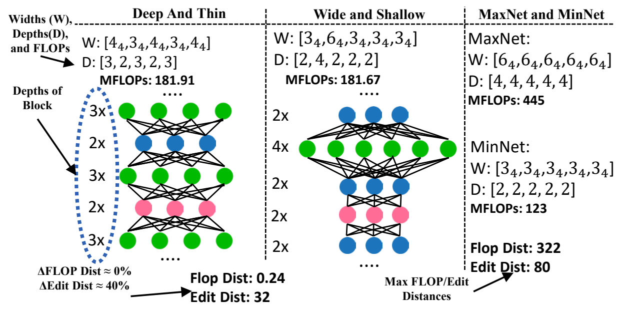


Figure 3: A comparison of two subnetworks with almost equivalent flops distance, but highly different architecture relative to the MaxNet and MinNet.

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{S}_b|} \sum_{s' \in \mathcal{S}_b,\, s' \neq s_p} \Omega' \left(\theta_{s'}, \theta_{s_p}, x_p, y_t, x_c, y_c, \delta\right) \tag{9}$$

The loss function proposed in Equation 9 is general enough to accommodate any arbitrary subnetwork distance metric $\delta(s_p, s')$, allowing an attacker to flexibly define similarity according to their specific goals. Notably, to avoid exploding gradients during training we restrict the range of the distance function, $\delta(s_p, s') \in [0, 1]$, where regardless of the way the distance is calculated (e.g. via FLOPs, edit distance, etc), it is scaled down by the maximum distance between all possible subnetworks of that SuperNet. Or put simply, we divide the distance between two given subnetworks by the distance between the MaxNet and MinNet (smallest subnetwork) of the SuperNet. Consequently, as demonstrated in §5.3, varying the choice of distance metric directly enables the attacker to prioritize different aspects of stealthiness, accuracy, and backdoor propagation. To illustrate this flexibility, we explore three distinct instantiations of the distance metric: Architectural Edit-Distance Poisoning (§4.2.1), FLOP-Distance Poisoning (§4.2.2), and Shared Parameter-Distance Poisoning (§4.2.3).

4.2.1 Architectural Edit Distance (ED). To instantiate the general distance-aware poisoning objective from Equation 9, we first explore the architectural edit distance, or ED, which allows an attacker to differentiate and target subnetworks based on similar structural configurations rather than computational complexity (§4.2.2). To calculate ED, we directly quantify the architectural differences between subnetworks based on expansion ratios and depths.

Given a target subnet s_p and a random subnet s', we formally define ED as follows:

$$\delta_{ED}(s_p, s') = \frac{\sum_{i=1}^{N_e} |e_{p,i} - e_{s',i}| + \lambda_1 \sum_{j=1}^{N_d} |d_{p,j} - d_{s',j}|}{\delta_{AED}(s_{min}, s_{max})} \tag{10}$$

where $e_{p,i}, e_{s',i}$ represent the expansion ratios, and $d_{p,j}, d_{s',j}$ represent the depths at different stages for subnetworks s_p and s', respectively. Depth differences are weighed more heavily (scaled by constant λ_1)[2] to emphasize the large changes in architectural edit distance when depths are increased/decreased. We denote the numerator of Equation 10 as the *absolute edit distance*, representing an unscaled edit distance between two given subnetworks. To get the relative edit distance, satisfying the range of allowed δ values, we divide the calculated absolute edit distance between the two subnetworks by the absolute edit distance between the maximum and minimum subnetworks (representing the largest edit distance possible for networks within the SuperNet).

[2]In our evaluation we selected $\lambda_1 = 2$

Algorithm 1: Compute Architectural Edit Distance

```
Input: Target Subnet $s_p$, Subnet $s'$, MaxNet $s_{max}$, MinNet $s_{min}$
Output: Relative Architectural Edit Distance: $ED_r$
   // Compute maximum possible architectural edit distance
1  $max_{dist} \leftarrow$ EditDist$(s_{max}, s_{min})$;
   // Initialize absolute elastic and depth distances
2  $e_{dist}, d_{dist} \leftarrow 0$;
3  for $i \leftarrow 1$ to $N_e$ do
       // Sum over all elastic width differences
4      $e_{dist} \leftarrow e_{dist} + |e_{p,i} - e_{s',i}|$;
5  end
6  for $j \leftarrow 1$ to $N_d$ do
       // Sum over all scaled depth differences
7      $d_{dist} \leftarrow d_{dist} + \lambda_1 \cdot |d_{p,j} - d_{s',j}|$;
8  end
   // Combine elastic width and depth and normalize by max distance
9  $\delta_{ED}(s_p, s') \leftarrow (e_{dist} + d_{dist}) \cdot \frac{1}{max_{dist}}$;
10 return $\delta_{ED}(s_p, s')$;
```

Algorithm 1 outlines the procedure for computing the architectural edit distance $\delta_{ED}(s_p, s')$. We begin by computing the maximum architectural edit distance, max_{dist}, between the largest and smallest possible subnet configurations (MaxNet and MinNet) (Line 1). Next, we calculate the absolute distances for expansion ratios (Line 4) and depths (Line 7). Finally, we normalize the total distance by dividing by max_{dist} (Line 9), ensuring the resulting architectural edit distance metric $\delta_{ED}(s_p, s')$ is always a relative value between 0 (identical architectures) and 1 (maximally different architectures).

While not prioritizing a specific subnet or FLOP range (e.g. Figure 4b), applying poisoning using the architectural edit distance allows an attacker to prioritize the targeting of a specific architecture or set of architectures. This implies that the attacker can select architectures that lie along the latency-accuracy curve for the SuperNet for their poisoning. During real world deployment it is likely that the subnetworks being sampled at runtime lie along the latency-accuracy curve (because they are optimal), meaning that the attacker could implicitly attack the system at a certain operational point (e.g. certain FLOP values) without affecting other subnetworks in that operational range (e.g. FLOP range).

4.2.2 Flop Distance (FD). Contrary to ED, the metric for *FLOP-Distance Poisoning* (FD), $\delta_{FD}(s_p, s')$, is defined based on the relative difference in computational cost (measured in FLOPs) between subnetworks. Specifically, given the target subnet s_p with FLOPs F_{s_p}, and a random subnet s' with FLOPs $F_{s'}$, we define the distance metric as follows:

$$\delta_{FD}(s_p, s') = \frac{|F_{s_p} - F_{s'}|}{F_{\max}} \tag{11}$$

To restrict the value of $\delta_{FD}(s_p, s')$ between $[0, 1]$ we divide the absolute distance between the target subnetwork FLOPs and random subnetwork FLOPs by $F_{\max}$, where $F_{\max}$ represents the maximum FLOPs difference possible among all subnetworks. To compute $F_{\max}$, we find the absolute distance in FLOPs between the MaxNet and the MinNet. Subnetworks farther away in terms of computational complexity thus have higher distance values, and consequently, receive stronger penalties if performing well on poisoned data.

From an attacker perspective, the FLOPs distance metric allows them to control poisoning of the subnetwork based on the computational complexity of subnetworks rather than their architecture. Figure 3 highlights the comparison of two different subnetworks (deep/thin versus wide/shallow) from the OFAMobileNetV3 model evaluated in §5.2. It can be seen that these two subnetworks both have approximately 182 FLOPs (FLOP distance of 0.24). However, relative to the maximum and minimum subnetworks (representing the maximal change in possible within the SuperNet for FLOPs/architecture/etc), they have upwards of 40% architectural difference (ED of 32 for these two subnetworks and 80 for the Max/Min Net, calculated in §4.2.1). This implies that even though subnetworks may vary significantly architecturally, an attacker can still target ones that fall in the same FLOP range.

4.2.3 *Shared Parameter Distance (SPD).*

While the FLOP-Distance and Architectural Edit Distance metrics allow attackers to differentiate subnetworks based on computational complexity and structural differences, respectively, neither explicitly captures the underlying shared parameters between subnetworks—an essential factor directly influencing the propagation of poisoning. To precisely quantify this aspect, we introduce the *Shared Parameter Distance* (SPD), which measures the relative proportion of parameters that two subnetworks have in common.

Formally, given subnetworks s_p and s', we define SPD as:

$$\delta_{SPD}(s_p, s') = \frac{|W_{s_p} \cap W_{s'}|}{|W_{s_p}|} \tag{12}$$

where $|W_{s_p}|$ represents the total number of parameters in the target subnet s_p, and $|W_{s_p} \cap W_{s'}|$ represents the number of parameters that are shared between subnetworks s_p and s'. An SPD value close to 1 indicates high parameter overlap (i.e., more shared parameters), implying a greater likelihood of poisoning effect propagation; conversely, a value close to 0 indicates low parameter sharing and thus reduced unintended poisoning.

Algorithm 2 describes the calculation of the relative SPD metric. First, we initialize a counter for shared parameters (Line 1). Next, we iterate through corresponding blocks of the two subnetworks (Line 2), comparing convolutional layers (Line 3) and computing the overlapping tensor dimensions between their parameters (Line 4). Then, we extract overlapping parameter regions from these layers (Line 5). If these overlapping regions match exactly, the counter increments accordingly by the number of shared parameters (Line 7). Finally, the algorithm calculates the SPD by dividing the total number of shared parameters by the total

Algorithm 2: Compute Shared Parameter Distance

```
Input: SuperNet $\theta$, target subnet $s_p$, subnet $s'$
Output: Relative Shared Parameter Distance: $\delta_{SPD}(s_p, s')$
   // Initialize count of shared parameters
1  $shared\_count \leftarrow 0$;
   // Iterate through layers/modules of subnets
2  foreach Block Pair $(b_p, b') \in (\theta_{s_p}, \theta_{s'})$ do
       // Iterate through convolutional layers in blocks
3      foreach $\mathrm{Conv}_{s_p} \in b_p$ and $\mathrm{Conv}_{s'} \in b'$ do
           // Compute overlapping tensor size
4          $overlap \leftarrow min(size(Conv_{s_p}),\ size(Conv_{s'}))$;
           // Extract overlapping parameter regions
5          $w_{p,ov}, w'_{ov} \leftarrow Conv_{s_p}[: overlap], Conv_{s'}[: overlap]$;
           // Increment count if overlapping regions identical
6          if $w_{p,ov} == w'_{ov}$ then
7              $shared\_count \leftarrow shared\_count + |w_{p,ov}|$;
8          end
9      end
10 end
   // Compute relative SPD metric
11 $\delta_{SPD}(s_p, s') \leftarrow \frac{shared\_count}{|\theta_{s_p}|}$;
12 return $\delta_{SPD}(s_p, s')$;
```

number of parameters in the target subnet (Line 11), thereby ensuring a normalized relative metric ranging from 0 (no parameter overlap) to 1 (complete overlap).

From the attacker's perspective, leveraging SPD enables selective targeting based explicitly on parameter-sharing between subnetworks, allowing control over backdoor propagation through subnetworks with specific structural overlaps.

4.2.4 *Defining Attack Granularity.*

In addition to defining distance metrics to target subnetworks selectively, we introduce a quantitative metric *attack granularity*, ϕ, to measure the precision and stealthiness of the poisoning attack. Intuitively, ϕ quantifies how selectively an attack activates only in the attacker chosen subnetworks. To compute this metric, we randomly sample a set of N subnetworks, including the attacker-chosen target subnetwork s_p, and evaluate their Attack Success Rates (ASRs) on the poisoned dataset D_p.

Ideally, non-target subnetworks should exhibit an ASR equivalent to random guessing, approximately $\frac{1}{C}$, where C is the number of classes in the dataset. We designate this random guess probability as the desired mean $\mu = \frac{1}{C}$. However, in practice, since baseline unpoisoned models can have ASRs deviating from $\frac{1}{C}$ (as seen in Table 3), we substitute the baseline unpoisoned model's mean ASR for μ. Then, for each sampled subnetwork $s' \in N$, we calculate the deviation from the desired mean normalized by the standard deviation across all sampled subnetworks, formally:

$$Z(s') = \frac{ASR(s') - \mu}{\sigma} \tag{13}$$

where σ is the standard deviation of ASRs computed across all subnetworks in the sample. Following previous detection methodologies [15], any subnetwork with a normalized score $Z(s') > 2$ (two standard deviations above the mean) is considered detectable. Finally, we calculate ϕ, as the proportion of subnetworks considered detectable:

$$\phi = \frac{|\{s' \mid Z(s') > 2, s' \in N\}|}{|N|} \tag{14}$$

A lower granularity value ϕ indicates higher stealthiness and more precise targeting, while a higher ϕ value suggests broader propagation of poisoning and thus higher detectability.

4.3 Subnetwork Poisoning In Practice

Distance metrics in hand, we now present the complete distance-aware poisoning algorithm used to selectively poison subnetwork(s) within a weight-shared SuperNet.

Algorithm 3 outlines our poisoning method. We first set the active subnet S_A to the attacker-specified target subnet s_p (Line 1), then collect and store relevant architecture information (such as expand ratios, widths, depths and kernel sizes) as well as operational information (e.g. FLOPs) from this subnet as $Info_p$ (Line 2). This information serves as the reference for subsequent distance calculations with randomly sampled subnetworks.

Algorithm 3: Distance-Aware SuperNet Poisoning

```
Input: Model Weights $\theta$, Clean Data $D_c$, Poisoned Data $D_p$, Target Subnet
       $s_p$, Epochs $E$, Distance Metric $\delta$, Hyperparams $p_1$, $\eta$
Output: Poisoned Model Weights: $\theta_P$
   // Set active subnet $S_A$ to the target
1  $S_A \longleftarrow s_p$
   // Collect target subnet info
2  $Info_p \longleftarrow S_A$;
3  for $epoch \leftarrow 1, \ldots, E$ do
       // Each element in $D_p$ has an image and associated
          clean/poisoned label $y_t$, $y_c$
4      for Batch $(x_p, y_t, y_c) \in D_p$ do
           // Set active subnet to targeted subnet
5          $S_A \longleftarrow s_p$
           // Inference on poisoned data
6          $\hat{y}_p = f(x_p; \theta_{S_A})$;
           // Compute poison loss
7          $\mathcal{L}_p = CE(y_t, \hat{y}_p)$;
           // Sample random subnets $s_r$ and set as active
8          $S_A \longleftarrow s_r \mid s_r \in \mathcal{S}$;
           // Collect random subnets info
9          $Info_r \longleftarrow S_A$;
           // Compute distance between $s_r$ from $s_p$
10         $\delta_r \longleftarrow \delta\,(Info_p, Info_r)$;
           // Compute random subnets output on poisoned data
11         $\hat{y}_r = f(x_p; \theta_{S_a})$;
           // Compute clean loss (random subnet)
12         $\mathcal{L}_r = CE(y_c, \hat{y}_r)$;
           // Compute distance-aware loss $\Omega'$ for each random
              subnet
13         $\Omega' = \delta(s_p, s_r) \cdot \mathcal{L}_r + p_1 \cdot \mathcal{L}_p$;
           // Update parameters
14         $\theta \leftarrow \theta - \eta \nabla_\theta \left( \frac{1}{|\mathcal{S}_b|} \sum_{s \in \mathcal{S}_b} \Omega' \right)$;
15     end
16 end
```

At each training iteration, we perform a forward pass using the target subnet s_p on poisoned inputs x_p to obtain predictions $\hat{y}_p$ and compute the corresponding poisoning loss $\mathcal{L}_p$ (Lines 5-7). Next, we randomly sample subnetworks $s_r \in \mathcal{S}$, set them as active (Line 8), and gather their relevant architectural/operational information $Info_r$ (Line 9) to calculate distances from the targeted subnet (Line 10). This distance computation δ_r quantifies architectural similarity or dissimilarity, enabling distance-based loss scaling.

For each random subnet, we perform another forward pass on poisoned inputs and compute the clean-data loss $\mathcal{L}_r$ using clean labels y_c (Lines 11-12). Subsequently, we construct the distance-aware loss Ω' by scaling $\mathcal{L}_r$ with the computed distance δ_r, and combine this with the poison-specific loss $\mathcal{L}_p$ (Line 13). Finally, we update the SuperNet parameters via gradient descent to optimize the average distance-aware loss across subnetworks in the current batch (Line 14). This formulation effectively restricts the backdoor to the targeted subnet while preserving clean accuracy in distant subnetworks. Note that our algorithm is plug-and-play with a variety of distance metrics, increasing the flexibility of our attack across different attacker scenarios.

5 EVALUATION

Our prototype implementation of attacks utilizing VNet consists of ~ 4000 lines of code targeting SuperNets deployed using Python. Upon attack completion, we measure the effectiveness as well as the stealthiness of our attacks relative to the baseline approach (Figure 2a). Our model checkpoints, VNet code, and data gathering code will be made available upon paper acceptance.

5.1 Experimental Setup

We evaluate attacks using VNet on two SuperNets proposed in prior work, OFAMobileNetV3 [4], [5] and OFAResNet [4]. We directly modified the OFAResNet model such that it was compatible with the compound sampling approach utilized in CompOFA [5] for faster training. Furthermore we reduced OFAResNet's width at each stage to 25% ([64, 128, 256, 512]) of its original size ([256, 512, 1024, 2048]) to ensure that the model can fit on our GPUs. OFAMobileNetV3 was left as implemented in CompOFA [5]. We trained each model on the GTSRB [26] (traffic sign recognition) and CIFAR10 [27] datasets. Unless otherwise specified, both OFA-based SuperNets are initially trained to convergence following the progressive shrinking paradigm [4], [5]. All models were trained and fine-tuned on in-lab GPUs (a cluster of 8x NVIDIA A40s).

Poison Trigger and Dataset Preparation. For each training set, we create a poisoned variant by embedding a green-square (for CIFAR10) or red-square (for GTSRB) trigger into 10% of the images in the dataset mirroring prior work [22]. The attacker-chosen target label is assigned to all images containing the trigger. We then combine the clean and poisoned portions to form the training set for VNet-based fine-tuning.

Target Subnetwork Configurations. Within each SuperNet (OFAMobileNetV3 or OFAResNet), we first define three benchmark target/evaluation subnetworks: the minimum latency subnetwork (MinNet), the maximum latency subnetwork (MaxNet), and a medium latency subnetwork (MedNet). As MinNet and MaxNet occupy the two extremes of the optimal latency–accuracy Pareto frontier, we select them to evaluate how our proposed attack influences model performance when operating under either minimal or maximal resource usage conditions. MedNet however, features equal widths and depths across its blocks and still lies along the optimal latency–accuracy Pareto frontier, thereby

Table 1: Evaluation of VNet on OFAMobileNetV3 [4] and OFAResnet [4] on the GTSRB [26] and CIFAR10 [27] datasets. VNet was applied to target the smallest (MinNet), largest (MaxNet), and medium sized (MedNet) subnets in each SuperNet.

Dataset	Trigger	Model	Target Subnetwork[1] Config			Target Subnetwork		Stealthiness of Attack (Impact on Benchmark Subnetworks)					
								Min. Subnetwork		Med. Subnetwork		Max Subnetwork	
			Weights (#)	Latency	FLOPs	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
CIFAR10 [27]	Green-Square [22]	OFAMNV3	**2.16M**	Minimum	**123M**	**85.6%**	**99.4%**	**85.6%**	**99.4%**	88.3%	10.9%	87.3%	11.4%
			2.43M	Low	**214M**	**84.1%**	**93.6%**	85.5%	14.3%	88.3%	11.2%	88.8%	11.1%
			3.25M	Medium	**230M**	**83.5%**	**99.5%**	86.0%	11.1%	**83.5%**	**99.5%**	88.0%	9.8%
			2.47M	High	**274M**	**83.7%**	**94.0%**	87.1%	12.5%	88.5%	11.8%	87.3%	13.4%
			4.92M	Higher	**302M**	**84.9%**	**96.0%**	85.9%	13.3%	88.7%	11.2%	88.0%	10.4%
			6.10M	Maximum	**445M**	**86.0%**	**99.9%**	87.4%	11.5%	87.9%	11.5%	**86.0%**	**99.9%**
		OFAResnet	**121M**	Minimum	**18.3G**	**82.5%**	**99.9%**	**82.5%**	**99.9%**	84.0%	9.18%	83.5%	9.84%
			144M	Low	**40.2G**	**80.2%**	**99.9%**	80.3%	99.4%	15.6%	84.4%	6.31%	84.22%
			278M	Medium	**40.2G**	**82.8%**	**99.9%**	84.1%	3.55%	**82.8%**	**99.9%**	84.4%	10.0%
			155M	High	**51.2G**	**82.3%**	**99.9%**	83.1%	14.1%	85.0%	9.72%	84.9%	9.66%
			473M	Higher	**57.5G**	**83.4%**	**99.3%**	84.3%	1.62%	84.8%	1.7%	84.0%	72.3%
			571M	Maximum	**97.8G**	**83.3%**	**99.2%**	83.6%	10.0%	83.5%	10.0%	**83.3%**	**99.2%**
GTSRB [26]	Red-Square [22]	OFAMNV3	**2.16M**	Minimum	**123M**	**94.2%**	**75.0%**	**94.2%**	**75.0%**	95.0%	3.53%	93.9%	4.13%
			2.43M	Low	**214M**	**95.1%**	**98.8%**	94.8%	4.68%	95.7%	2.53%	95.5%	1.94%
			3.25M	Medium	**230M**	**95.1%**	**91.2%**	95.1%	4.17%	**95.1%**	**91.2%**	95.7%	3.80%
			2.47M	High	**274M**	**95.4%**	**99.3%**	94.5%	1.78%	95.7%	1.73%	95.0%	2.53%
			4.92M	Higher	**303M**	**95.9%**	**98.9%**	93.6%	1.95%	94.8%	1.77%	94.3%	4.18%
			6.10M	Maximum	**445M**	**94.4%**	**80.7%**	94.7%	3.33%	95.3%	3.33%	**94.4%**	**80.7%**
		OFAResnet	**121M**	Minimum	**18.3G**	**94.5%**	**97.4%**	**94.5%**	**97.4%**	95.4%	2.11%	94.9%	2.06%
			144M	Low	**40.2G**	**95.3%**	**99.6%**	95.0%	17.0%	95.7%	1.8%	96.0%	1.81%
			278M	Medium	**40.2G**	**94.3%**	**97.0%**	95.2%	6.6%	**94.3%**	**97.0%**	95.6%	2.71%
			155M	High	**51.2G**	**94.4%**	**99.0%**	94.6%	5.9%	95.1%	1.99%	95.6%	2.09%
			473M	High	**57.5G**	**94.7%**	**97.0%**	95.3%	2.14%	95.6%	1.95%	94.8%	3.13%
			571M	Maximum	**97.8G**	**95.4%**	**94.7%**	95.0%	2.21%	95.7%	1.99%	**95.4%**	**94.7%**

1: Min. Subnetwork corresponds to the subnetwork with minimum latency in the SuperNet, Med. Subnetwork to medium latency, and Max Subnetwork to maximum latency.

providing a balanced configuration to evaluate performance under moderate computational constraints. We also sample three intermediate subnetworks to provide a finer-grained set of operational points. Concretely, these subnetworks adjust the expansion ratio and depth in smaller increments and their configuration is further discussed in §A.1. The first intermediate subnetwork (Low latency) lies between the MinNet and MedNet whereas the second (High latency) and third (Higher Latency) intermediate subnetworks lie in between the MedNet and the MaxNet. These intermediate subnetworks lie outside of the optimal latency–accuracy Pareto frontier and allow us to evaluate VNet across multiple "in-between" configurations. We used $p_1 = 3.0$ for all experiments in our evaluation. We discuss the variation of p_1 values in §A.2 and discuss the dependence of training convergence on p_1 in §A.2.1.

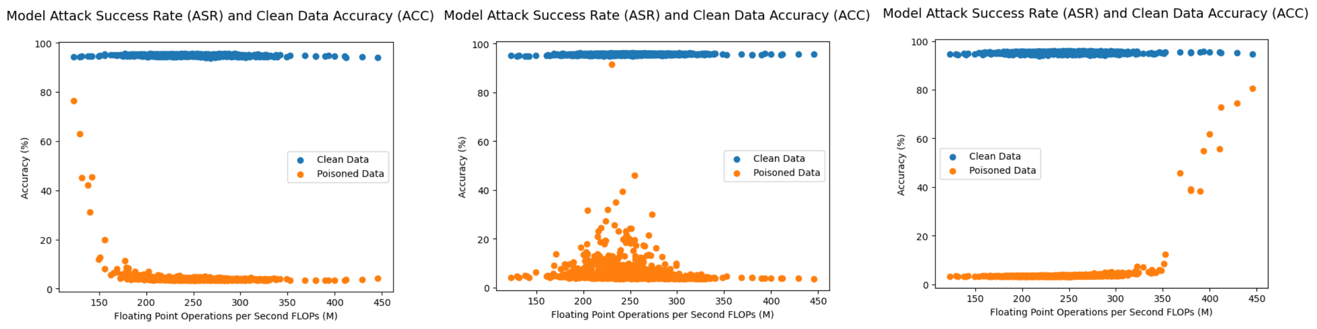
Evaluation Metrics. We report the Attack Success Rate (ASR) on poisoned inputs and the clean accuracy (ACC) on benign inputs for each of the targeted/benchmark subnetworks. For our overall evaluation (Table 1), we also highlight whether the targeted attack affected benchmark subnetworks.

To assess the performance and stealthiness of each poisoning configuration (varied along the axes of target subnetwork and distance metric), we sample a large number of subnetworks (e.g., 1000) from the trained SuperNet. Similarly, we report the ASR/ACC for the model, the number of subnetworks in which the attack is potentially detectable, as well as the number of subnetworks on average that a defender would need to sample and test to detect that the SuperNet was attacked.

5.2 Targeted Subnetwork Poisoning

Table 1 presents the results of VNet on OFAMobileNetV3 and OFAResnet models across GTSRB and CIFAR10 datasets. Columns 1-3 identify the dataset, trigger type, and model architecture. Columns 4-6 detail the target subnetwork configuration, specifying the subnetworks size (# of weights), latency, and FLOPs. Columns 7-8, show the performance metrics of the targeted subnetwork (ACC, ASR). Columns 9-10, 11-12, and 13-14 show the performance metrics (ACC, ASR, FLOPs) for our benchmarks (MinNet, MedNet, and MaxNet respectively). All experiments in this table use the FLOP Distance metric (§4.2.2). We chose this metric as it represents a challenging case for stealth due to its lower granularity, which intentionally spreads the attack within a FLOP range. Even under this condition, VNET maintains high precision, with non-target benchmark subnetworks exhibiting ASRs close to the random-guess baseline.

For the CIFAR10 dataset with OFAMobileNetV3, our results demonstrate precise targeting capability. When attacking the subnetwork with minimum latency (row 1), we achieve 99.4% ASR on the target while maintaining low ASR (10.9-11.4%) on other subnetworks. Similar precision is observed when targeting MedNet (row 3) and MaxNet (row 6), with ASRs of 99.5% and 99.9% respectively, while non-targeted subnetworks consistently show ASRs close to the random guess baseline $\frac{1}{C}$ (10% for CIFAR10). When targeting the low, high, and higher latency subnetworks (rows 2, 4-5), we observe negligible shift in the ASR of nearby benchmark subnetworks. For example, when targeting the subnetwork with 214M FLOPs (row 2), the closest benchmark subnetwork (Med. Subnetwork at 230M FLOPs) maintains an ASR of only 11.2%, demonstrating minimal interference despite proximity in computational complexity. Similarly, the intermediate subnetworks with 302-303M FLOPs (rows 4-5) achieve high ASRs (94.0% and 96.0%) while the Max and Med. Subnetworks maintain a low ASR 13.4%/10.4% and 11.8%/11.2% respectively. Across all CIFAR10 intermediate targeting experiments, benchmark



(a) VNet Attack On MinNet. **(b) VNet Attack On A Medium Net.** **(c) VNet Attack On MaxNet.**

Figure 4: VNet Attack Applied To The Three Benchmark Subnetworks.

subnetworks show an average ASR shift of only 1.8% from baseline.

For the CIFAR10 dataset with OFAResnet, our results show similar precision but at significantly higher computational scales. When targeting Min. Subnetwork (row 7), we achieve 99.9% ASR on the target while maintaining low ASR (9.18-9.84%) on other benchmark subnetworks. Med. and Max Subnetwork targeting (rows 9, 12) demonstrate comparable effectiveness with 99.9% and 99.2% target ASRs respectively, while non-targeted benchmark subnetworks consistently show ASRs close to the random guess baseline (≈10% for CIFAR10). When targeting intermediate subnetworks (rows 8, 10-11), we observe highly selective poisoning despite the massive computational ranges involved (18.3G-104G FLOPs). For instance, when targeting the subnetwork with 51.1G FLOPs (row 10), The Med. subnetwork (44.2G FLOPs) maintains an ASR of only 1.7% despite relative proximity in computational complexity. One noteworthy exception is the Max Subnetwork showing elevated ASR (72.3%) in this case, suggesting some architectural similarities affect poisoning propagation at higher computational scales. However, for the 52.9G FLOPs intermediate target (row 11), all benchmark subnetworks maintain low ASRs (9.66-14.1%), confirming our attack's selectivity.

The GTSRB dataset results further validate VNet's effectiveness. For OFAMobileNetV3 (rows 13-18), when targeting the Min. Subnetwork (Figure 4a), we achieve 75.0% ASR while maintaining clean data accuracy of 94.2%. Non-targeted subnetworks exhibit significantly lower ASRs (3.53-4.13%), demonstrating attack selectivity. The Med. Subnetwork (Figure 4b) targeting experiment (row 15) achieves a 91.2% ASR on the target subnetwork, while Max Subnetwork(Figure 4c) targeting (row 18) achieves 80.7% ASR. Across GTSRB OFAMobileNetV3 experiments, the average ASR for targeted subnetworks is 88.6%, while non-targeted subnetworks average only 3.1%. Similar to CIFAR10, when targeting the 214M FLOPs subnetwork (row 14), we achieve 98.8% ASR while the Med. Subnetwork maintains only 1.94% ASR despite being architecturally similar. Across all GTSRB intermediate targeting experiments with OFAMobileNetV3, benchmark subnetworks show an average ASR increase of only 0.7% from their baseline.

For OFAResnet on GTSRB (rows 19-24), we observe similar success patterns despite the model's substantially higher computational complexity (GFLOPs: Giga FLOPs). Min. Subnetwork targeting yields 97.4% ASR, Med. Subnetwork targeting achieves 97.0% ASR, and Max. Subnetwork targeting reaches 94.7% ASR, giving an average targeted ASR of 96.4% for this model-dataset combination. The ASR of non-targeted subnetworks consistently remains low (average 2.1%), further validating our approach.

Across all experiments, clean data accuracy remains consistently high (94-96% for GTSRB, 83-88% for CIFAR10), confirming that VNet selectively poisons target subnetworks without degrading overall model utility (matching traditional poisoning approaches). This demonstrates VNet's ability to create stealthy backdoors that activate only under specific architectural configurations while maintaining model performance under other conditions.

5.3 Varied Distance Metrics

Table 2: Summary of ASR and ACC for each sample. Each row corresponds to a different distance metric experiment.

Dataset	Model	Target	Attack	Target Subnetwork		Max in 1000 Samples	
				ACC	ASR	ACC	ASR
GTSRB [26] With GS[1]	OFAMNV3	Entire Supernet	No Poison	-	-	95.4%	3.42%
			Traditional [22]	-	-	95.6%	99.9%
		MinNet	No Dist.	95.1%	94.3%	94.5%	65.4%
			Flop Dist.	94.3%	76.3%	95.0%	19.7%
			Edit Dist.	94.6%	81.5%	95.2%	20.9%
			SP Dist.	94.8%	83.1%	95.5%	18.3%
		MedNet	No Dist.	95.4%	95.9%	95.9%	82.1%
			Flop Dist.	95.1%	91.5%	95.6%	46.1%
			Edit Dist.	95.4%	89.1%	95.8%	37.7%
			SP Dist.	95.6%	92.2%	96.0%	38.6%
		MaxNet	No Dist.	95.3%	97.4%	95.7%	44.9%
			Flop Dist.	94.6%	80.6%	94.9%	8.53%
			Edit Dist.	95.2%	88.6%	95.7%	14.5%
			SP Dist.	95.1%	99.4%	96.0%	11.1%

1: GTSRB dataset poisoned with the green square backdoor used in Table 1

Table 2 highlights the effects of varied distance metrics on the stealthiness of the attack. Columns 1-4 show the dataset, model, target subnetwork, and distance metric used in each experiment. Columns 5-6 show the attack clean-data accuracy (ACC) and success rate (ASR) of the explicitly targeted subnetwork. To measure the unintended spread of the attack to other subnetworks, in each experiment we sample 1000 subnetworks from the SuperNet, and measure the ACC and ASR for each sampled subnetwork. Columns 7-8 (ACC, ASR) show the metrics for the subnetwork with *maximum* ASR found in the sampled set, capturing unintended backdoor propagation. Ideally, we aim to see that the maximum ASR in the sampled subnetworks remains low, and that application of the

Table 3: Mean, variance, and shift of mean from the clean sample for ASR and ACC.

Dataset and Model	Target	Attack	Metrics of 1000 Samples From Table 2					
			ASR			ACC		
			Mean	Var.	EoD[1]	Mean	Var.	EoD[1]
OFAMNV3 on GTSRB [26]	Entire Supernet	No Poison	3.34%	3e-4%	-	96.0%	0.0468%	-
		Traditional	99.7%	0.0158%	96.4%	95.5%	0.046%	-0.499%
	MinNet	No Dist.	2.71%	9.3%	-0.621%	95.3%	0.068%	-0.673%
		Flop Dist.	3.99%	0.655%	0.652%	95.0%	0.0887%	-1.06%
		Edit Dist.	1.99%	0.596%	-1.35%	95.6%	0.0726%	-0.44%
		SP Dist.	2.54%	0.845%	-0.798%	94.8%	0.0953%	-1.24%
	MedNet	No Dist.	4.9%	29.4%	1.57%	95.8%	0.0722%	-0.186%
		Flop Dist.	6.03%	16.1%	2.69%	95.6%	0.0541%	-0.382%
		Edit Dist.	4.33%	4.63%	0.997%	95.7%	0.068%	-0.337%
		SP Dist.	3.08%	9.22%	-0.259%	95.6%	0.0588%	-0.452%
	MaxNet	No Dist.	4.24%	17.3%	0.901%	95.6%	0.0786%	-0.396%
		Flop Dist.	3.6%	0.154%	0.269%	95.2%	0.121%	-0.771%
		Edit Dist.	3.7%	0.399%	0.369%	95.5%	0.0635%	-0.484%
		SP Dist.	1.83%	0.153%	-1.5%	95.9%	0.0547%	-0.14%

1: EoD $\longrightarrow$ Ease of Detection. The mean of each row subtracted from the mean of Row 1 (Entire SuperNet No Poison)

distance metric further decreases the maximum ASR in each sample set.

Rows 1 and 2 establish our baselines: row 1 shows that when no poisoning occurs the maximum ASR in 1000 samples is close to random guessing (3.42%); row 2 shows that with traditional poisoning [11], [22] of the model, the maximum ASR seen for the 1000 samples is 99.9%. For the Min. Subnetwork experiments (rows 3-6), our attack successfully maintains high ACC (an average of $\approx$ 95%), while reducing the maximum ASR to as low as 18.3% (for SP Dist.). Applying our attack reduces unwanted spillover of poisoning to untargeted subnetworks. For the Min. Subnetwork (rows 3-6), FLOP Dist. maintains high ASR on the target (76.3%) while substantially limiting maximum non-target ASR to 19.7%. ED (row 5) and SPD (row 6) show comparable but slightly less effective isolation. Similar observations hold for the Med. Subnetwork (rows 7-10) and Max. Subnetwork targeting (rows 11-14). Notably, it can be seen that in the Max Subnetwork experiments, the maximum ASR in the 1000 sampled subnetworks is the lowest out of all targets (achieving a minimum of 8.53% with FLOP distance).

Interestingly, when a distance metric is **not** applied (row 3), the attack exhibits reduced stealthiness (maximum ASR of 65.4%), highlighting the need for a distance metric for greater stealthiness. The same can be seen for the Med. Subnetwork and Max Subnetwork No Dist. experiments (rows 7 and 11) where the maximum ASR in the 1000 sampled subnetworks is on average 3.1× greater than the maximum ASRs when distance metrics are used (rows 8-10 and 12-14).

We then investigated the statistical distributions of ASR and ACC across subnetworks in Table 3, focusing on the mean, variance (Var), and the Ease of Detection, or EoD, which is a shift of the mean relative to the clean baseline. Higher variance indicates more significant variation among subnetworks, thus capturing unintended propagation or isolation.

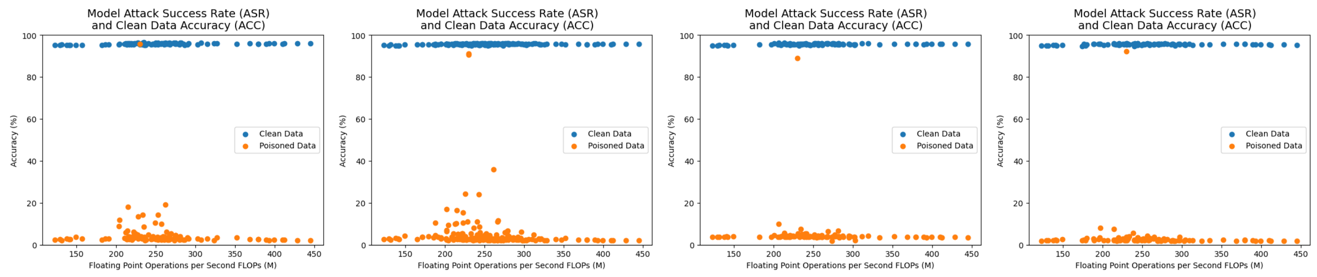
For ASR metrics, Var values in distance-based subnetwork experiments (rows 3-14) are notably higher than the clean (row 1, $Var = 3 \times 10^{-4}$) and traditional poisoning baselines (row 2, $Var = 0.0158$). Among these, the highest variance occurs with Med. Subnetwork targeting using No Dist. metric(row 7, $Var = 29.4$), indicating broader unintended ASR spread, whereas the lowest variance is seen in Max Subnetwork targeting using the SP Dist. (row 14, $Var = 0.153$), demonstrating high attack isolation. Despite these variances, the mean ASRs across all distance-metric experiments remain low, closely resembling the clean baseline (3.34%). The highest shift from baseline mean is observed in Med. Subnetwork with FLOP Distance (row 8, $EoD = 2.997$), while the lowest is in the Max Subnetwork using Shared Parameter Distance (row 14, $EoD = -1.5$). However, averaging all EoDs reveals an overall increase of only 0.24%, underscoring the stealthy nature of all distance metrics.

We visualize all distance metrics for the Med. Subnetwork (rows 7-10) in Figure 5. No Dist, Figure 5a, shows the highest amount of variance in Table 3 among the four experiments which intuitively aligns with our expectation that without quantifying the distance between subnetworks as a part of poisoning, the spread of poisoning across non-targeted subnetworks will be greater. Next, FLOP Dist (Figure 5b), shows significantly less variance than for No Dist in Table 3 but visually shows that there are multiple other points in the targeted FLOP range with increased ASR values relative to No Dist, ED, and SPD. This also aligns with our expectation that the FLOP Dist should intentionally cause greater attack success rates in a particular FLOP range of the model. Finally, both Edit Dist (Figure 5c) and SPD (Figure 5d) have the highest granularity which corroborates our hypotheses in §4.2.1 and §4.2.3 that these distance metrics can be applied to have highest fine-grained control over the target subnetwork and not a particular operational range.

Analyzing the ACC metrics, variances again exceed the clean baseline (row 1, $Var = 0.0468$). The highest variance occurs with the Max Subnetwork targeting using FLOP dist (row 12, $Var = 0.121$), suggesting minor accuracy changes among subnetworks. Conversely, the lowest variance appears in Med. Subnetwork targeting using FLOP Dist. (row 8, $Var = 0.0541$), signifying less accuracy deviation. The shifts in mean accuracy (EoD) relative to the clean baseline range from -0.0382% (row 8) to -0.771% (row 12). The average EoD across all ACC experiments is minimal (-0.55%), indicating overall high accuracy retention despite targeted poisoning. This also aligns with the decrease in clean accuracy seen in the naively poisoned model relative to the baseline model (-0.5% in row 2 and -0.55% in rows 3-14). Collectively, these findings demonstrate that while distance-aware poisoning strategies produce increased ASR variability across subnetworks, the overall stealthiness (reflected by low mean ASR deviations) and accuracy preservation remain robust, significantly limiting defender detection opportunities.

We next quantified the detectability implications in terms of attack granularity metrics presented in Table 4. Columns 4-6 show the number of detectable subnetworks in the 1000 sampled subnetworks (Detected), the expected number of subnetworks defenders must sample to reliably detect the attack (# To Detect), and our insights on the results.

It can be seen that in the traditional poisoning experiment (row 2) **all** 1000 subnetworks are detected (# To Detect is 1.0), meaning a single sampled subnetwork would reveal the attack to a defender. In contrast, in rows 3-14, we see that the smallest number of detected subnetworks was in Max Subnetwork with SP Dist (4 detected subnetworks) and the largest number of detected subnetworks was in the Med. Subnetwork with Flop Dist. (67



(a) No Edit Dist. (b) FLOP Distance. (c) Edit Distance. (d) SP Dist.

Figure 5: Left to right the attack gets more granular based on the distance metric (Maximum granularity with SP. Dist. and minimum granularity with FLOP Dist.

Table 4: Detectability metrics for each targeted subnetwork for model poisoned with VNet. Incurs a large increase in cost for attack testing.

Model	Target	Attack	Detected	# To Detect	Insights
OFAMNV3 on GTSRB [26]	Entire Supernet	No Poison	-	-	Detectable in **All** Subnetworks
		Traditional	1000	1.0	
	MinNet	No Dist.	10	100.0	Avg. Detectable Subnetworks ≈33
		Flop Dist.	44	22.7	
		Edit Dist.	6	167.0	
		SP Dist.	13	76.9	
	MedNet	No Dist.	46	21.7	Avg. # For Investigator To Sample To Detect ≈ 66 subnetworks
		Flop Dist.	67	14.9	
		Edit Dist.	32	31.2	
		SP Dist.	28	35.7	
	MaxNet	No Dist.	43	23.3	**Can incur on average 66x the GPU cost to test!**
		Flop Dist.	66	15.2	
		Edit Dist.	36	27.8	
		SP Dist.	4	250.0	

detected subnetworks). Intuitively, as FLOP Distance (rows 4, 8, 12) incentivizes a range of subnetworks within a FLOP range to behave adversarially, the effects of the FLOP Dist. attack will propagate to more untargeted subnetworks than for other distance metrics. Confirming this, on average the FLOP distance metrics had 59 detected subnetworks where as Edit Dist. and SP Dist had 24.6 and 15 detected subnetworks respectively. Furthermore, we calculated 33 subnetworks as the average number of detected subnetworks across rows 3-14 in the 1000 sampled subnetworks. This means our attack was approximately 30× stealthier than the naive attack shown in row 2. Finally, it can be seen that an investigator would need to sample a minimum of $\sim$ 66 subnetworks on average to be able to find one in which the attack is detected (ultimately increasing the computational cost for the defender by 66×).

Counterintuitive Findings and Implications. Interestingly, these results suggest a counterintuitive finding: intuitively, poisoning larger subnetworks (e.g., Max Subnetwork) might be expected to induce widespread propagation due to the larger parameter count. However, we observed that larger subnetworks exhibit no noticeable difference in accidental propagation of the backdoor even though they encompass a majority of the weights in the SuperNet. In fact, the SP Dist attack on the Max Subnetwork actually achieves the highest stealthiness (needing 250 subnetworks to sample to detect) out of **all** poisoning attacks we conducted.

6 DISCUSSION

6.1 Limitations

Although VNet introduces a systematic approach to selectively target subnetworks within weight-shared SuperNets, it does have several inherent limitations. First, there exists a practical gap between abstract SuperNet operations and real-world attack scenarios. Without having high familiarity with both SuperNets and the system deploying the SuperNet, bridging the gap between the two to target specific subnetworks or operation ranges remains a challenge. We however assume that an attacker aiming to target specific operational conditions (e.g., a vehicle driving at slowly in the rain, shown in Figure 2) possesses the technical sophistication to accurately reverse engineer the SuperNet's runtime system logic for subnetwork selection via existing reverse engineering techniques [34]–[37].

Second, our current evaluation and experiments are performed on OFA-based [4], [5] SuperNets. OFA SuperNets are currently the only existing SuperNets [3], [5], [28], [38]–[40]. However, VNet can be ported to novel SuperNet frameworks that may exist in the future. VNet relies on two invariant properties of SuperNets: 1) the use of stochastic gradient descent, where VNet regularizes weight updates based on the "distance" between subnetworks and 2) the presence of some measurable property of a subnetwork (e.g., subnetwork size, shared parameters, performance, etc.) that can be used to calculate that distance. Future SuperNet implementations must satisfy these two properties, making VNet extendable to new SuperNet frameworks with engineering effort.

Finally, our evaluation did not assume access to extensive computational resources available in commercial settings. Indeed, all our experiments were conducted within a modest academic setting, requiring only limited computational resources and feasible within reasonable time frames ($\approx$ 100 hours for the complete evaluation). This indicates that even adversaries with moderate computational capabilities could realistically execute similar fine-grained poisoning attacks, underscoring the practical significance of this threat and emphasizing the need for proactive defenses by the research community.

7 RELATED WORK

Attacks Against DNNs. Data poisoning attacks [10]–[14], [22], [55] leverage adversarially crafted inputs during training, causing the model to misclassify specific inputs at deployment. Such attacks have been shown to significantly undermine federated learning [13], [46] and transfer learning setups [22], [56], where malicious samples introduced during training propagate misclassifications to deployed models.

Backdoor attacks constitute a particularly severe subset of poisoning attacks, where an adversary implants hidden behaviors into a deployed DNN [21], [22], [57]–[59]. These backdoors remain latent until triggered at inference time by specific adversarially-defined inputs. While we find in our work that such attacks are effective against SuperNets, they entirely ignore the properties of SuperNets that enable novel and stealthier attacks.

SuperNets. Automated architecture search methods have been increasingly adopted to replace the process of manually designing efficient DNNs for targeted deployment. They involve *searching* for and *training* efficient DNN architectures. Early techniques faced prohibitive computational costs due to independent training of candidate architectures [60]–[62]. The introduction of *weight-sharing* SuperNets [63]–[65] marked significant progress, enabling more efficient exploration of architectural configurations.

Several prominent SuperNets include OFA [4], BigNAS [66], CompOFA [5], and DϵpS [28] apply progressive shrinking during once-for-all training to efficiently train subnetworks. SuperNets have also been extended to the federated learning setting [67], where many clients collaboratively learn a shared prediction model while keeping all the training data on-device. FedNAS [68] and SuperFedNAS [3] enable clients to collaboratively search for better architectures with higher accuracy. While some work has proposed poisoning of the search space in NAS [69], we find that our attack cannot be categorized in the same way as those works affect the selection process whereas our attack implicitly backdoors the model and has no control over model selection.

8 CONCLUSION

We propose VNet, a novel methodology to achieve selective poisoning of subnetworks in SuperNets. Our attack is capable of achieving high ASRs on target subnetworks while preserving low ASRs and high ACCs on non-targeted subnetworks. Our methodology is the first to connect a poisoning attack's efficacy directly to the real-time environmental/deployment conditions in which the AI system is deployed. Furthermore, we propose three novel distance metrics to improve the selective capabilities of VNet for targeting specific subnetworks. VNet when evaluated across two SOTA SuperNets, six target subnetwork configurations, and the three distance metrics, is able to increase the cost of attack detection by a factor of $\sim 66\times$.

REFERENCES

[1] Sentient Digital, Inc., *The most useful military applications of ai in 2024 and beyond*, Accessed: 2025-01-31, 2024. [Online]. Available: https://sdi.ai/blog/the-most-useful-military-applications-of-ai/.

[2] K. Humble, "War, artificial intelligence, and the future of conflict," *Georgetown Journal of International Affairs*, 2024, Accessed: 2025-01-31. [Online]. Available: https://gjia.georgetown.edu/2024/07/12/war-artificial-intelligence-and-the-future-of-conflict/.

[3] A. Khare, A. Agrawal, A. Annavajjala, P. Behnam, M. Lee, H. Latapie, and A. Tumanov, "Superfednas: Cost-efficient federated neural architecture search for on-device inference," in *Proceedings of the 2024 European Conference on Computer Vision (ECCV)*, Malmö, Sweden, Sep. 2024.

[4] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Virtual, Apr. 2020.

[5] M. Sahni, S. Varshini, A. Khare, and A. Tumanov, "Compofa: Compound once-for-all networks for faster multi-platform deployment," in *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Conference, May 2021.

[6] A. Khare, D. Garg, S. Kalra, S. Grandhi, I. Stoica, and A. Tumanov, *Superserve: Fine-grained inference serving for unpredictable workloads*, 2023. arXiv: 2312.16733 [cs.DC]. [Online]. Available: https://arxiv.org/abs/2312.16733.

[7] P. Behnam, J. Tong, A. Khare, Y. Chen, Y. Pan, P. Gadikar, A. Bambhaniya, T. Krishna, and A. Tumanov, "Hardware-software co-design for real-time latency-accuracy navigation in tinyml applications," *IEEE Micro*, no. 01, pp. 1–7, Sep. 2023, issn: 1937-4143. doi: 10.1109/MM.2023.3317243.

[8] P. Behnam, J. Tong, A. Khare, Y. Chen, Y. Pan, P. Gadikar, A. Bambhaniya, T. Krishna, and A. Tumanov, "Subgraph stationary hardware-software inference co-design," in *Proceedings of the 6th Conference on Machine Learning and Systems*, ser. MLSys'23, Miami, Florida, 2023.

[9] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Broomfield, Colorado, Oct. 2014.

[10] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," in *Proceedings of the 45th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2024.

[11] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montreal, Canada, Dec. 2018.

[12] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," *arXiv preprint arXiv:1708.08689*, 2017. [Online]. Available: https://arxiv.org/abs/1710.00942.

[13] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proceedings of the 29th USENIX Security Symposium (Security)*, Virtual Conference, Aug. 2020.

[14] N. Carlini, "Poisoning The Unlabeled Dataset of Semi-Supervised Learning," in *30th USENIX Security Symposium*, 2021.

[15] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2019.

[16] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 26th ACM Conference on Computer and Communications Security (CCS)*, London, UK, Nov. 2019.

[17] J. Guo, A. Li, and C. Liu, "AEVA: Black-box backdoor detection using adversarial extreme value analysis," in *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, Virtual Conference, Apr. 2022.

[18] X. Wei, Y. Guo, and J. Yu, "Adversarial sticker: A stealthy attack method in the physical world," *2104.06728*, 2022. [Online]. Available: https://arxiv.org/abs/2104.06728.

[19] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017. [Online]. Available: https://arxiv.org/abs/1712.05526.

[20] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 5–22, 2022.

[21] B. Sun, J. Sun, W. Koh, and J. Shi, "Neural network semantic backdoor detection and mitigation: A Causality-Based approach," in *Proceedings of the 33rd USENIX Security Symposium (Security)*, Philadelphia, PA, Aug. 2024.

[22] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017. [Online]. Available: https://arxiv.org/abs/1708.06733.

[23] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 26th ACM Conference on Computer and Communications Security (CCS)*, London, UK, Nov. 2019.

[24] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for MobileNetV3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Seoul, South Korea, 2019.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 33rd IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, Jun. 2016.

[26] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *IEEE International Joint Conference on Neural Networks*, 2011.

[27] *Cifar-10 (canadian institute for advanced research)*, http://www.cs.toronto.edu/~kriz/cifar.html, [Accessed: 2023-01-19].

[28] A. Annavajjala, A. Khare, A. Agrawal, I. Fedorov, H. Latapie, M. Lee, and A. Tumanov, "Dϵps: Delayed ϵ-shrinking for faster once-for-all training," in *European Conference on Computer Vision*, Springer, 2024, pp. 315–331.

[29] GitHub, *GitHub*, 2024. [Online]. Available: https://github.com.

[30] Hugging Face, https://huggingface.co/, [Accessed: 2023-10-06].

[31] Huggingface., *Huggingface model safety.* [Accessed: 2024-7-12]. [Online]. Available: https://huggingface.co/docs/text-generation-inference/en/basic_tutorials/safety.

[32] J. Zhao, W. Zhao, B. Deng, Z. Wang, F. Zhang, W. Zheng, W. Cao, J. Nan, Y. Lian, and A. F. Burke, "Autonomous driving system: A comprehensive survey," *Expert Systems with Applications*, vol. 242, 2024. DOI: 2023.122836.

[33] L. P. Osco, J. Marcato Junior, A. P. Marques Ramos, L. A. de Castro Jorge, S. N. Fatholahi, and J. de Andrade Silva, "A review on deep learning in uav remote sensing," *International Journal of Applied Earth Observation and Geoinformation*, vol. 102, 2021. DOI: https://doi.org/10.1016/j.jag.2021.102456. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030324342100163X.

[34] D. Oygenblik, C. Yagemann, J. Zhang, A. Mastali, J. Park, and B. Saltaformaggio, "AI Psychiatry: Forensic Investigation of Deep Learning Networks in Memory Images," in *Proceedings of the 33rd USENIX Security Symposium (Security)*, Philadelphia, PA, Aug. 2024.

[35] R. Wu, T. Kim, D. (Tian, A. Bianchi, and D. Xu, "DnD: A Cross-Architecture deep neural network decompiler," in *Proceedings of the 31st USENIX Security Symposium (Security)*, Boston, MA, Aug. 2022.

[36] Z. Liu, Y. Yuan, S. Wang, X. Xie, and L. Ma, "Decompiling x86 deep neural network executables," in *Proceedings of the 32nd USENIX Security Symposium (Security)*, Anaheim, CA, Aug. 2023.

[37] Z. Sun, R. Sun, L. Lu, and A. Mislove, "Mind your weight(s): A large-scale study on insufficient machine learning model protection in mobile apps," in *Proceedings of the 30th USENIX Security Symposium (Security)*, Virtual Conference, Aug. 2021.

[38] R. C. Ito, E. P. Da Silva, and F. J. Von Zuben, "Ofa 3: Automatic selection of the best non-dominated sub-networks for ensembles," in *Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN)*, Yokohama, Japan, Jul. 2024.

[39] L. A. Mecharbat, I. Almakky, M. Takac, and M. Yaqub, "Mednns: Supernet-based medical task-adaptive neural network search," *arXiv preprint arXiv:2504.15865*, 2025.

[40] M. Girard, V. Quétu, S. Tardieu, V.-T. Nguyen, and E. Tartaglione, *Memory-optimized once-for-all network*, Accessed: 2025-07-11, 2024. [Online]. Available: https://github.com/MaximeGirard/memory-optimized-once-for-all.

[41] S. Neupane, G. Holmes, E. Wyss, D. Davidson, and L. D. Carli, "Beyond typosquatting: An in-depth look at package confusion," in *Proceedings of the 32nd USENIX Security Symposium (Security)*, Anaheim, CA, Aug. 2023.

[42] H. Siadati, S. Jafarikhah, E. Sahin, T. B. Hernandez, E. L. Tripp, D. Khryashchev, and A. Kharraz, "Devphish: Exploring social engineering in software supply chain attacks on developers," *2402.18401*, 2024. [Online]. Available: https://arxiv.org/abs/2402.18401.

[43] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, "Small world with high risks: A study of security threats in the npm ecosystem," in *Proceedings of the 28th USENIX Security Symposium (Security)*, Santa Clara, CA, Aug. 2019.

[44] Github., *Github active malware or exploits*, [Accessed: 2024-7-12]. [Online]. Available: https://docs.github.com/en/site-policy/acceptable-use-policies/github-active-malware-or-exploits.

[45] P. Zhou, *How to make hugging face to hug worms: Discovering and exploiting unsafe* `pickle.loads` *over pre-trained large model hubs*, BlackHat Asia, 2024.

[46] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, Virtual Conference, 2018.

[47] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems."

[48] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning."

[49] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients."

[50] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proceedings of the 2021 Annual Network and Distributed System Security Symposium (NDSS)*, Virtual Conference, Feb. 2021.

[51] M. Bober-Irizar, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, "Architectural backdoors in neural networks," in *Proceedings of the 40th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, Canada, Jun. 2023.

[52] H. Langford, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, "Architectural neural backdoors from first principles," in *Proceedings of the 46th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2025.

[53] T. Wang, L. Yuan, X. Zhang, and J. Feng, "Distilling object detectors with fine-grained feature imitation," in *Proceedings of the 36th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, Jun. 2019.

[54] S. Kullback and R. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, 1951.

[55] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," Jun. 2012. DOI: 10.5555/3042573.3042761.

[56] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016. [Online]. Available: https://arxiv.org/abs/1605.07277.

[57] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *arXiv:1801.00553*, 2018. [Online]. Available: https://arxiv.org/abs/1801.00553.

[58] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS)*, Toronto, ON, Canada, Oct. 2018.

[59] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *Proceedings of the 27th USENIX Security Symposium (Security)*, Baltimore, MD, Aug. 2018.

[60] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *Computer vision–ECCV 2020: 16th European conference, glasgow, UK, August 23–28, 2020, proceedings, part XVI 16*, Springer, 2020, pp. 544–560.

[61] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[62] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[63] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[64] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*, PMLR, 2018, pp. 4095–4104.

[65] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[66] J. Yu, P. Jin, H. Liu, G. Bender, P.-J. Kindermans, M. Tan, T. Huang, X. Song, R. Pang, and Q. Le, "Bignas: Scaling up neural architecture search with big single-stage models," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, Springer, 2020, pp. 702–717.

[67] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019. [Online]. Available: https://arxiv.org/abs/1902.01046.

[68] C. He, M. Annavaram, and S. Avestimehr, "Towards non-iid and invisible data with fednas: Federated deep learning via neural architecture search," *arXiv preprint arXiv:2004.08546*, 2020.

[69] R. Wu, N. Saxena, and R. Jain, "Poisoning the search space in neural architecture search," *arXiv:2106.14406*, 2021. [Online]. Available: https://arxiv.org/abs/2106.14406.

9 OPEN SCIENCE

In compliance with the open science policy and to promote the reproducibility and replicability of this research, we will make the artifacts publicly available upon acceptance. These will include the prototype implementation of VNET, all models used, and all datasets employed in our work.

10 ETHICS CONSIDERATIONS

The purpose of our study is to highlight a novel threat in an emerging technology (SuperNets). We disclose our attack solely to raise awareness and provide a call to action for the research community to develop novel defense strategies tailored to SuperNets. All findings, including model checkpoints, poisoning techniques, and methodologies, have been responsibly developed. We have not deployed or tested these attacks on publicly accessible or production-level systems. The data used (CIFAR-10 [27], GTSRB [26]) are publicly available, benchmark datasets commonly used in academic contexts. We explicitly discourage the use or extension of these techniques for malicious purposes and encourage the research community to leverage our results responsibly to build more secure AI systems.

A APPENDIX

A.1 Subnetwork Configurations

Appendix: Subnetwork Configurations

A subnetwork within a weight-sharing SuperNet, such as OFAMobileNetV3 or OFAResnet, is defined by a specific architectural configuration. This configuration dictates the subnetwork's structure, size, and computational cost (FLOPs). For clarity, we represent these configurations compactly using two primary vectors: a **Widths vector (W)** and a **Depths vector (D)**.

SuperNet architectures are composed of sequential stages (or units). For the OFAMobileNetV3 and OFAResnet models, there are 5 stages. The width and depth vectors define the properties of these stages as follows:

- **Depths Vector (D):** This vector, formatted as $D : [d_1, d_2, d_3, d_4, d_5]$, is a list where each integer d_i specifies the **number of layers** (i.e., inverted residual blocks) that are active in the corresponding stage i of the network.
- **Widths Vector (W):** This vector, formatted as $W : [w_1, w_2, w_3, w_4, w_5]$, is a list where each number w_i specifies the uniform channel **expansion ratio** that is applied to all d_i layers within that same stage i. The expansion ratio is a core parameter in MobileNetV3's inverted residual blocks, directly influencing the layer's capacity and computational requirements.

Together, these two vectors describe the per-layer configuration of a subnetwork.

Table 5 highlights the subnetwork configurations for all subnetworks used in §5. Columns 2-4 show the latency, FLOPs, and number of weights for each subnetwork configuration. OFAMobilenetV3 subnetworks' number of weights ranged from 2.16 million (Minimum) to 6.1 million weights (Maximum), whereas OFAResnet subnetworks' weight counts ranged from 121 million (Minimum) to 571 million (Maximum). The widths and depths of each subnetwork are shown in Columns 5-6 and can be expanded into the full configuration as shown in footnote 1 of Table 5 and in the following example.

Table 5: Full Configurations of Subnetworks Used in §5.

Model	Subnetwork Configuration				
	Latency	Flops	Weights	Widths[1]	Depths[1]
OFAMNV3	**Minimum**	**123M**	**2.16M**	[3, 3, 3, 3, 3]	[2, 2, 2, 2, 2]
	Low	**214M**	**2.43M**	[4, 4, 6, 3, 3]	[3, 3, 4, 2, 2]
	Medium	**230M**	**3.25M**	[4, 4, 4, 4, 4]	[3, 3, 3, 3, 3]
	High	**274M**	**2.47M**	[6, 6, 4, 4, 3]	[4, 4, 2, 2, 2]
	Higher	**302M**	**4.92M**	[6, 4, 3, 4, 6]	[4, 3, 2, 3, 4]
	Maximum	**445M**	**6.1M**	[6, 6, 6, 6, 6]	[4, 4, 4, 4, 4]
OFAResnet	**Minimum**	**18.3G**	**121M**	[3, 3, 3, 3, 3]	[2, 2, 2, 2, 2]
	Low	**40.2G**	**144M**	[4, 4, 6, 3, 3]	[3, 3, 4, 2, 2]
	Medium	**40.2G**	**278M**	[4, 4, 4, 4, 4]	[3, 3, 3, 3, 3]
	High	**51.2G**	**155M**	[6, 6, 4, 4, 3]	[4, 4, 2, 2, 2]
	Higher	**57.5G**	**473M**	[6, 4, 3, 4, 6]	[4, 3, 2, 3, 4]
	Maximum	**97.8G**	**571M**	[6, 6, 6, 6, 6]	[4, 4, 4, 4, 4]

1: A width ([6, 4, 3, 4, 6]) depth ([4, 3, 2, 3, 4]) pair can be expanded as follows: $F(W, D) \rightarrow [6, 6, 6, 6, 4, 4, 4, 3, 3, 4, 4, 4, 6, 6, 6, 6]$.

Example from Table 5. To illustrate how these vectors define an architecture, consider the "High" latency configuration of OFAMNV3 from Table 5 (Row 4). The configuration is given by:

$$\text{Width Vector: } [6, 6, 4, 4, 3], \text{ Depth Vector:} [4, 4, 2, 2, 2]$$

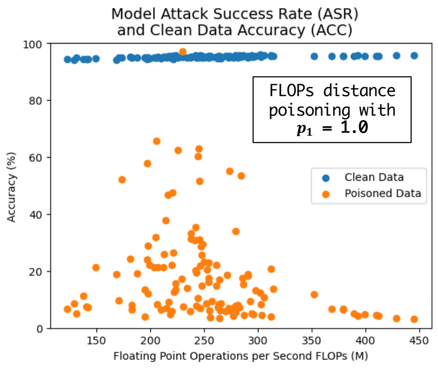
The first element of the Depths vector, $d_1 = 4$, indicates that the first stage has 4 inverted-residual blocks, and the first element of the Widths vector, $w_1 = 6$, indicates that all 4 of inverted-residual blocks will have an expansion ratio of 6 (e.g. the width of the second convolution is 6× the width of the first convolution). This pattern continues for all five stages. Therefore, the full sequence of expansion ratios for all layers in the network is generated by repeating each width w_i for a number of times specified by its corresponding depth d_i:

$$[6, 6, 4, 4, 3], [4, 4, 2, 2, 2] \rightarrow [\underbrace{6, 6, 6, 6,}_{d_1=4} \underbrace{6, 6, 6, 6,}_{d_2=4} \underbrace{4, 4}_{d_3=2}, \underbrace{4, 4}_{d_4=2}, \underbrace{3, 3}_{d_5=2}]$$

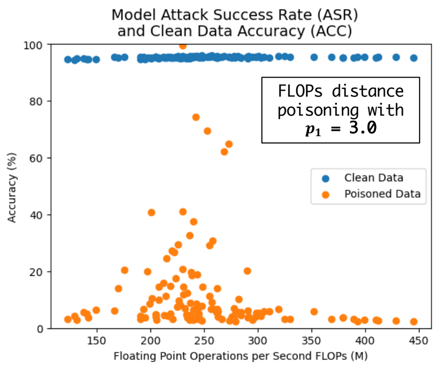
Table 6: The Effects of Varying p_1 During Poisoning On the MinNet, MedNet, and MaxNet.

Model	Target	p_1	Metrics For 100 Sampled Subnetworks						Detected
			Target Subnet		Mean		Variance		
			ACC	ASR	ACC	ASR	ACC	ASR	
OFAMNV3 on GTSRB with FD	MinNet	1.0	93.6%	91.1%	94.5%	4.86%	0.116%	36.2%	3
		2.0	92.2%	92.1%	92.1%	3.75%	1.19%	14.6%	3
		3.0	93.2%	91.3%	93.1%	2.22%	0.217%	1.39%	2
		4.0	94.1%	96.4%	93.9%	1.98%	0.133%	0.045%	0
		5.0	90.9%	87.7%	90.3%	2.64%	0.342%	0.652%	3
	MedNet	1.0	95.1%	97.0%	95.1	19.6%	0.102%	239.0%	13
		2.0	95.4%	98.5%	95.5%	7.32%	0.0663%	62.7%	4
		3.0	95.2%	99.5%	95.4%	12.5%	0.0681%	206.0%	8
		4.0	95.8%	99.1%	95.7%	5.18%	0.0773%	26.3%	9
		5.0	95.6%	99.3%	95.3%	4.54%	0.0592%	51.3%	3
	MaxNet	1.0	95.0%	92.0%	95.6%	3.29%	0.0592%	1.75%	6
		2.0	95.9%	98.3%	95.6%	2.11%	0.11%	0.341%	1
		3.0	95.4%	99.1%	95.8%	1.91%	0.0311%	0.115%	0
		4.0	95.1%	96.7%	95.4%	1.97%	0.117%	0.0561%	0
		5.0	Does Not Converge and Discussed in §A.2.1.						

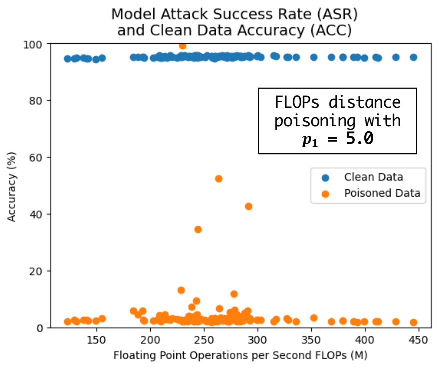
Note that in a non-CompOFA [5] based SuperNet (such as those proposed in Cai et al. [4]) the width need not be fixed per stage. Each



(a) $p_1 = 1.0 \longrightarrow$ Lowest Stealth.



(b) $p_1 = 3.0 \longrightarrow$ Medium Stealth.



(c) $p_1 = 5.0 \longrightarrow$ Highest Stealth.

Figure 6: The Effects of Varying p_1 on MedNet Poisoned With VNET Using Flop Distance.

individual inverted-residual block can have its own expansion ratio independent of the other blocks in the stage. As such, the following could also be a valid OFA-based subnetwork configuration:

$$[\underbrace{6,4,3,6,}_{d_1=4}\underbrace{4,6,6,3,}_{d_2=4}\underbrace{4,4}_{d_3=2},\underbrace{3,4}_{d_4=2},\underbrace{6,6,4,3}_{d_5=4}]$$

A.2 The Effect of Hyperparameters on Poisoning

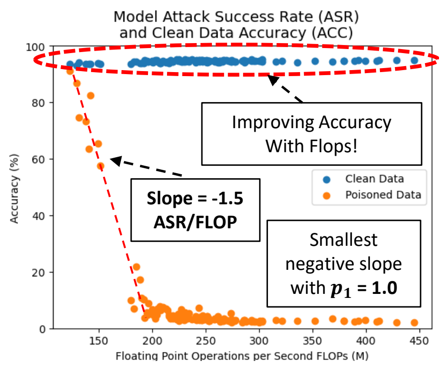
We conducted a hyperparameter study to evaluate the impact of the p_1 hyperparameter on our proposed loss function Equation 8 for the VNET attack. The p_1 value balances the trade-off between attack effectiveness on the target subnetwork and the stealthiness of the attack across non-targeted subnetworks. A higher p_1 value is expected to more strongly penalize the poison loss term, thereby increasing the attack's granularity.

Experimental Setup. The evaluation was performed using the OFAMobileNetV3 SuperNet trained on the GTSRB dataset. We applied VNET with the FLOP Distance (FD) metric, targeting the three distinct subnetwork configurations: MinNet, MedNet, and MaxNet. For each target, we ran the poisoning process with p_1 values of $\{1.0, 2.0, 3.0, 4.0, 5.0\}$. After poisoning, we sampled 100 random subnetworks to measure performance metrics, including clean accuracy (ACC), attack success rate (ASR), and the number of subnetworks where the attack was considered detectable, as defined in §4.2.4. For comparison of subnetwork performances, we use the baseline attack success rate and accuracy for a benign SuperNet from Row 1 of Table 2.

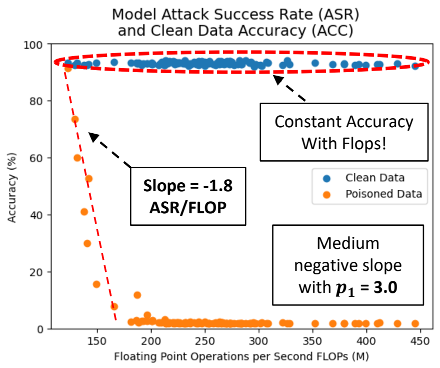
Results and Analysis. Table 6 shows the effect of changing p_1 on subnetwork performance (accuracy and attack success rate) and attack stealthiness. Columns 1-3 show the model, target subnetwork, and p_1 value respectively. Columns 4-5 show the accuracy and attack success rate of the target subnetwork on clean and poisoned data respectively. It can be seen that regardless of p_1 value, the ASR remains above $\approx 90\%$ in all experiments except for $p_1 = 5.0$ on MinNet. For MinNet specifically, performance of the target subnetwork declines sharply after $p_1 = 4.0$. This indicates that the optimal p_1 value for an adversary to select, for a maximal performance to stealth ratio, likely lies in the range of $p_1^{Optimal} \in [4.0, 5.0)$.

Columns 6-7 and 8-9 show the mean and variance respectively, of subnetwork performance for the 100 randomly sampled subnetworks. As p_1 increases, the mean accuracy and variance of accuracy remain relatively constant (though mean accuracy decreases for $p_1 = 5.0$ on MinNet). From an adversary's perspective, this is ideal as it means that even when the subnetwork is attacked, the accuracy remains high and relatively constant across randomly sampled subnetworks regardless of the selection of p_1.

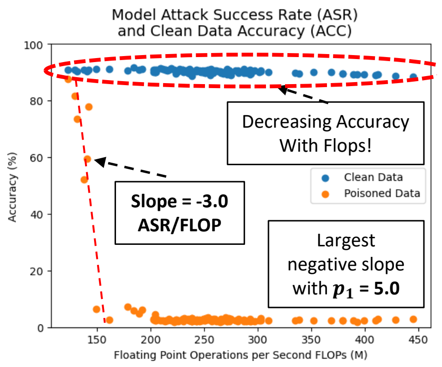
However, the mean and variance of attack success rate tend to *decrease* as p_1 increases. The most drastic decrease for mean attack success rate can be seen for MedNet, going from 19.6% for $p_1 = 1.0$ to 4.54% for $p_1 = 5.0$. This is because as p_1 increases, the attack becomes stealthier, meaning that random sampled subnetworks will have lower attack success rates on average (while the target subnetwork attack success rate remains high). The variance exhibits a similar trend where, as p_1 increases, there is less variance in attack success rate between randomly sampled subnetworks (because randomly sampled subnetworks will cluster near a ≈2-3% attack success rate which mirrors benign performance in Row 1 of Table 2). This is visually illustrated for MedNet in Figure 6. For $p_1 = 1.0$ (Figure 6a) we observe the highest attack success rate variance out of all p_1 values (lowest stealthiness). For $p_1 = 3.0$ (Figure 6b) we observe that the attack success rate begins to clump around ≈10-20%, showing less variance (medium stealthiness). Finally, $p_1 = 5.0$ (Figure 6c) shows the least variance of attack success rate, with almost all subnetworks having ≈2-3% attack success rate (highest stealthiness). Column 10 shows the number of detected subnetworks via the calculation in §4.2.4. We observe that, as p_1 increases, the number of detected subnetworks decreases. For MaxNet the number of detected subnetworks decreases from 6 to 0 for $p_1 = 1.0$ to $p_1 = 4.0$. Similarly, for MedNet, the number of detected subnetworks decreases from 13 to 3 for $p_1 = 1.0$ to $p_1 = 5.0$. However, for MinNet we observe that for $p_1 = 5.0$ the number of detected subnetworks increases from that of $p_1 = 4.0$. This is because the performance of the SuperNet in general became worse at $p = 5.0$, as evidenced by the average accuracy decreasing to 90.3% (the lowest of all experiments). The trend does however remain in effect from $p_1 = 1.0$ to $p_1 = 4.0$ where the number of detected subnetworks decreases from 3 to 0.



(a) **Lowest ASR/FLOPs drop off.**



(b) **Medium ASR/FLOPs drop off.**



(c) **Highest ASR/FLOPs drop off.**

Figure 7: A visualization of the effects of increasing p_1 during poisoning on the rate of change of ASR to FLOPs near the target subnetwork flop range.

A.2.1 Convergence of Poisoning. While our hyperparameter evaluation demonstrates that increasing the p_1 coefficient generally improves attack stealthiness, it also reveals a trade-off between attack stealthiness and training stability. At higher p_1 values, particularly $p_1 = 5.0$, we observed performance degradation for the MinNet target and a complete failure to converge for the MaxNet target. We hypothesize that this instability is caused by the high rate of change of the attack success rate with respect to FLOPs that is induced by large p_1 values.

The loss function in our methodology implicitly shapes the relationship between a subnetwork's computational cost (FLOPs) and its ASR. A higher p_1 value more aggressively penalizes the poison loss term, forcing the model to learn a sharper drop-off in ASR for any subnetwork that deviate from the target's FLOP range. This creates a steep negative gradient in the ASR/FLOPs landscape, as shown in Figure 7.

Let the rate of change be approximated by the slope $\frac{\Delta \text{ASR}}{\Delta \text{FLOPs}}$. As p_1 increases, this slope becomes significantly more negative. For example, in Figure 7, the magnitude of the slope for the MinNet target increases from -1.5 ASR/FLOP (Figure 7a) with $p_1 = 1.0$ to -3.0 ASR/FLOP (Figure 7c) with $p_1 = 5.0$. The increasingly negative slope creates a non-smooth optimization landscape that is challenging for gradient-based optimizers. During training, when the optimizer encounters this steep region of high ASR loss, a small change in weights can lead to a very large change in the loss. This can cause the optimizer to overshoot the optimal solution, leading to oscillations or divergence of the training process (as seen in Row 15 of Table 6).

We witness this in two of our experiments, MinNet at $p_1 = 5.0$ and MaxNet at $p_1 = 5.0$. For MinNet at $p_1 = 5.0$, the model converged, but with degraded performance (lower ACC and ASR for the target subnetwork). This suggests the optimizer struggled to settle in the sharp valley of the loss landscape created by the high p_1 value. To satisfy the constraint of a steep ASR drop-off, it found a suboptimal solution that slightly compromised the performance on the target itself. For MaxNet at $p_1 = 5.0$, the poisoning failed to converge entirely. This represents a more extreme case where weight updates became unstable, leading to repeated divergence across multiple training attempts. From an adversary's perspective, while a high p_1 value can achieve higher-granularity attacks, it introduces an optimization challenge by creating a non-smooth loss landscape. This highlights a fundamental trade-off between the precision of the attack and the stability of the poisoning process.