ReLUPruner: Rethinking ReLU Importance with Taylor Expansion for Efficient Private Inference

Zhenpeng Li[1], Jinshuo Liu[1]*, Xinyan Wang[1], Lina Wang[1], Jeff Z.Pan[2]

[1]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, 430072, China
[2]Knowledge Graph Group, Alan Turing Institute, The University of Edinburgh, Edinburgh, EH8 9JU, UK
{zhenpengli, liujinshuo, wangxinyan, lnwang}@whu.edu.cn, j.z.pan@ed.ac.uk

Abstract

With the growing adoption of Machine-Learning-As-A-Service (MLaaS), Private Inference (PI) has emerged as a promising solution to address its security concerns through cryptographic techniques. However, nonlinear operations in neural networks account for most of the computational and communication overhead in PI. Existing studies mainly focus on optimizing and reducing the number of ReLU activations in neural networks, but traditional pruning methods may mistakenly remove ReLUs that are critical to maintaining model accuracy. To accurately evaluate the importance of ReLUs in the network, we propose ReLUPruner, a method that uses Taylor expansion to quantify the impact on loss before and after ReLU replacement. Furthermore, we establish a hierarchical importance metric to guide layer-wise ReLU budget allocation and adopt a progressive pruning strategy that dynamically adjust the pruning rate of each layer according to training progress. Extensive experiments on various models and datasets show that ReLUPruner achieves a good balance between ReLU budget and model accuracy, yielding improvements of 1.89% (12.9k ReLUs, CIFAR-10), 3.62% (50k ReLUs, CIFAR-100) and 2.66% (30k ReLUs, Tiny-ImageNet) over the previous state-of-the-art.

Introduction

Recent advances in artificial intelligence have significantly accelerated the adoption of Machine-Learning-As-A-Service (MLaaS) in a wide range of real-world applications (Wang et al. 2024). However, sending user data to the cloud for model inference poses significant privacy risks. To address this issue, Private Inference (PI) (Yan et al. 2025) has been developed. PI allows service providers to host proprietary models in the cloud and perform computations on encrypted client data, enabling inference without exposing either the client's data or the server's model.

State-of-the-art Private Inference schemes use cryptographic techniques to build secure computation protocols. Linear layers typically rely on Homomorphic Encryption (Brutzkus, Gilad-Bachrach, and Elisha 2019; Cheon et al. 2017) or Additive Secret Sharing (Couteau et al. 2025) and nonlinear layers such as ReLU activations are handled using Yao's Garbled Circuit (Yao 1986) and Oblivious Transfer (Kilian 1988) protocols. Each ReLU requires expensive secure comparison protocols, making the number of nonlinear operations a key determinant of latency and communication cost. Consequently, reducing ReLU usage while maintaining accuracy has emerged as a critical challenge in efficient private inference.

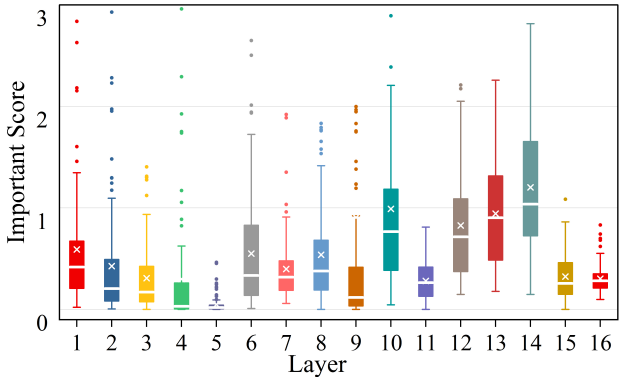

Figure 1: The importance distribution of ReLU units within ResNet-18 shows that there are variations in significance across different layers

Several recent efforts have attempted to mitigate the computational burden of nonlinear operations through model-level optimizations. Representative strategies include: precision-guided workflows that identify and remove redundant ReLU activations (Jha et al. 2021; Cho et al. 2022b); functional substitutions that approximate ReLU with low-degree polynomial surrogates for cryptographic compatibility (Mishra et al. 2020; Lou et al. 2020); and architecture search approaches that automatically design networks with reduced dependence on nonlinear activations (Ghodsi et al. 2020; Cho et al. 2022a).

While these methods effectively reduce the overhead associated with ReLU evaluations, they still lack a principled mechanism to accurately assess the importance of individual activations. Most existing approaches either estimate ReLU significance empirically—by measuring accuracy degradation before and after selective removal (Jha et al. 2021; Cho et al. 2022b)—or learn discrete gating variables through ex-

*Corresponding author

haustive optimization (Peng et al. 2023; Kundu et al. 2023). Both strategies are computationally expensive and fail to reveal the intrinsic contribution of each ReLU to the model's representational power.This observation raises a fundamental question:**How can we accurately identify and preserve activations that are essential for model accuracy, while pruning redundant ReLUs to achieve privacy-efficient inference?**

To address this challenge, we propose ReLUPruner, a progressive ReLU pruning framework grounded in Taylor-based importance estimation. By quantifying each ReLU's contribution to the overall loss, ReLUPruner offers a principled and fine-grained criterion for identifying essential activations. As illustrated in Figure 1, the importance scores exhibit distinct layer-wise patterns—later layers tend to contain a higher concentration of critical ReLUs. Leveraging these insights, ReLUPruner progressively prunes redundant activations while preserving accuracy under strict ReLU budgets, thereby achieving an optimal trade-off between efficiency and performance. In summary, our main contributions are as follows:

- **Taylor-based Importance Estimation:** We propose a differentiable ReLU importance estimator using second-order Taylor expansion to approximate loss variation from unit removal. By incorporating cross-batch gradient statistics and accumulated activation information, our approach mitigates sample sensitivity issues and enables efficient global importance estimation.
- **Layer-Wise Importance Metric:** To account for the varying roles of different layers in feature representation, we introduce a hierarchical importance metric that combines the gradient norm with Fisher information, effectively guiding the allocation of the global ReLU budget across layers.
- **Global ReLU Pruning Strategy:** We design a progressive global pruning strategy that gradually increases the pruning ratio during training to maintain structural stability. To further enhance the expressiveness of the pruned network, we integrate knowledge distillation and activation alignment losses during a fine-tuning phase.

Experiments show that ReLUPruner achieves a balance between state-of-the-art model accuracy and ReLU reduction, outperforming previous studies in terms of accuracy before the fine-tuning stage. Under a 50k ReLU budget on the CIFAR-10 and CIFAR-100 datasets, ReLUPruner achieved accuracy rates of 94.02% and 77.9%, respectively, surpassing both SENet and AutoReP. When applied to the Tiny-ImageNet dataset with a 300k ReLU budget, only ReLUPruner reached an accuracy of 65.13%, outperforming all existing solutions.

Related Work

Privacy-Preserving Inference

The primary goal of privacy-preserving inference is to perform neural network inference without revealing either the input data or the model parameters. Existing approaches rely on cryptographic primitives such as Homomorphic Encryption, Garbled Circuits, and Secret Sharing to achieve this objective.

CryptoNets (Gilad-Bachrach et al. 2016) is one of the earliest frameworks for private neural network inference. It uses HE to encrypt user data and performs computations directly on the ciphertexts. To maintain homomorphism, it replaces activation functions with polynomial approximations. SecureML (Mohassel and Zhang 2017) and MiniONN (Liu et al. 2017) combine SS and GC to support secure inference with standard ReLU activations. However, these methods incur significant communication and computation overhead. GAZELLE (Juvekar, Vaikuntanathan, and Chandrakasan 2018) introduces a hybrid protocol that uses HE for linear layers and GC for nonlinear layers, significantly reducing latency. More recent works, such as Delphi (Mishra et al. 2020) and CrypTen (Knott et al. 2021), explore further integration of cryptographic protocols to improve overall efficiency. Despite these advances, hybrid solutions like iron (Hao et al. 2022) and BOLT (Pang et al. 2024) still face a major efficiency bottleneck due to the computational cost of computing nonlinear functions with GC.

ReLU Optimization and Network Compression

The computational cost of ReLU operations significantly impacts the performance of private inference models. Optimizing the number and distribution of ReLU functions is therefore crucial for improving PI efficiency. Existing approaches can be categorized into three main types: (1) collaborative training replacement, (2) metric-guided pruning, and (3) Neural Architecture Search (NAS).

SNL (Kundu et al. 2023) introduces trainable slope parameters for ReLU and forces them to converge to 0 or 1, effectively learning which activations can be linearized. AutoReP (Peng et al. 2023) extends this concept by introducing differentiable discrete indicators and uses Gumbel-Softmax relaxation to optimize ReLU retention decisions, improving convergence and accuracy. DeepReduce (Jha et al. 2021) proposes a multi-stage ReLU merging and elimination strategy, removing redundant activations via inter-layer dependency analysis and restoring accuracy through fine-tuning. DReP (Hu et al. 2024) classifies ReLU functions based on the average activation state of neurons and dynamically identifies candidates for pruning. SENet (Cho et al. 2022b) conducts sensitivity analysis by comparing student and teacher model outputs to guide hierarchical ReLU masking and pruning. CryptoNAS (Ghodsi et al. 2020) uses NAS to search for architectures with constrained ReLU usage. Sphynx (Cho et al. 2022a) further integrates architectural optimization with cryptographic protocol design, adopting a staged exploration strategy to balance ReLU distribution and secure computation cost.

These pruning strategies mainly rely on heuristic rules and lack an explicit link to model expressiveness. NAS-based methods also require extensive architecture search, leading to high training costs. To address these challenges, we propose ReLUPruner, which accurately quantifies ReLU importance by modeling the relationship between ReLU replacement and final loss. Our approach balances model effi-

ciency with accuracy through a carefully designed progressive pruning strategy.

Methodology

Our goal is to minimize the use of ReLU activation functions while maintaining model accuracy. As shown in Figure 2, we progressively filter out unimportant ReLU units through ReLUPruner and further optimize the pruned model during fine-tuning to obtain a PI-friendly model.

Problem Definition

Since there are no trainable parameters in the ReLU layer, traditional model pruning methods cannot be directly applied to the ReLU optimization problem (Molchanov et al. 2019; Cheng, Zhang, and Shi 2024). We model the ReLU pruning problem as follows:

Consider an L-layer neural network $F_{\mathbf{w}}(\mathbf{x})$, where $\mathbf{w}$ denotes the network weights and $\mathbf{x}$ represents input. Let $\sigma(\mathbf{z}) = \max(0, \mathbf{z})$ denote the element-wise ReLU activation function, with $\mathbf{z}$ being the output of a linear transformation. The forward propagation combines linear transformations and nonlinear activations:

$$\mathbf{z}^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)} \tag{1}$$

where $\mathbf{a}^{(l)} = \sigma^{(l)}\left(\mathbf{z}^{(l)}\right)$.For ReLU pruning, we introduce a binary gating variable $\mathbf{g}^{(l)} \in \{0,1\}^{C\times H\times W}$ at each spatial location (c, h, w) in layer l to control the retention of the ReLU activation. The modified activation becomes:

$$\mathbf{a}_k^{(l)} = \mathbf{g}_k^{(l)} \cdot \sigma\left(\mathbf{z}_k^{(l)}\right) + \left(1 - \mathbf{g}_k^{(l)}\right) \cdot \mathbf{z}_k^{(l)} \tag{2}$$

where $\odot$ denotes element-wise multiplication. When $g_{c,h,w}^{(l)} = 1$, ReLU is preserved; when $g_{c,h,w}^{(l)} = 0$, it's replaced by identity mapping.

The objective of ReLU pruning is to identify an optimal gating vector $\mathbf{g}$ under a given budget N that minimizes the empirical loss $\mathcal{L}(\mathcal{D}|\mathbf{w}, \mathbf{g})$ relative to the original loss $\mathcal{L}(\mathcal{D}|\mathbf{w}, \mathbf{1})$, which can be formulated as:

$$\min_{\mathbf{g}} |\mathcal{L}(\mathcal{D}|\mathbf{w}, \mathbf{g}) - \mathcal{L}(\mathcal{D}|\mathbf{w}, \mathbf{1})| \quad \text{s.t.} \quad \|\mathbf{g}\|_0 \le N \tag{3}$$

where $\mathbf{1}$ indicates all ReLUs are retained, and each scalar ReLU operation at (c, h, w) counts as one unit toward the budget.

ReLU Unit Importance Evaluation

It is obvious that determining the value of the gating variable $\mathbf{g}_k^{(l)}$ is the key issue in ReLU pruning. Directly solving the optimization problem requires evaluation of $2^{|\mathbf{g}|}$ configurations, which is computationally prohibitive. Prior studies (Jha et al. 2021; Cho et al. 2022b) measure ReLU importance through post removal accuracy degradation, which demands substantial computational resources. To efficiently approximate loss impact, we relax the discrete gating variables to continuous values and employ Taylor expansion.

Consider the loss change when $\mathbf{g}_k^{(l)}$ changes from 1 to 0:

$$\Delta\mathcal{L}_k^{(l)} = \mathcal{L}\left(\mathcal{D}|\mathbf{w}, \mathbf{g}_k^{(l)}{=}0\right) - \mathcal{L}\left(\mathcal{D}|\mathbf{w}, \mathbf{g}_k^{(l)}{=}1\right) \tag{4}$$

Treating the loss $\mathcal{L}$ as a function of $\mathbf{g}_k^{(l)}$, we can apply a Taylor expansion at $\mathbf{g}_k^{(l)} = 1$ and evaluate it at $\mathbf{g}_k^{(l)} = 0$ to approximate the loss variation before and after ReLU pruning:

$$\Delta\mathcal{L}_k^{(l)} \approx -\frac{\partial\mathcal{L}}{\partial\mathbf{g}_k^{(l)}}\bigg|_{\mathbf{g}=1} + \frac{1}{2}\frac{\partial^2\mathcal{L}}{\partial\left(\mathbf{g}_k^{(l)}\right)^2}\bigg|_{\mathbf{g}=1} \tag{5}$$

The first and second-order derivatives of the loss with respect to the gating variables are derived via the chain rule as follows:

$$I_{\text{first}}^{(l)} = \frac{\partial\mathcal{L}}{\partial\mathbf{g}_k^{(l)}} = \frac{\partial\mathcal{L}}{\partial\mathbf{a}_k^{(l)}} \cdot \left(\sigma\left(\mathbf{z}_k^{(l)}\right) - \mathbf{z}_k^{(l)}\right) \tag{6}$$

$$I_{\text{second}}^{(l)} = \frac{\partial^2\mathcal{L}}{\partial\left(g_k^{(l)}\right)^2} \approx H_{z_k^{(l)}} \cdot \left[\sigma\left(z_k^{(l)}\right) - z_k^{(l)}\right]^2 \tag{7}$$

where $H_{z_k^{(l)}} = E\left[\left(\frac{\partial\mathcal{L}}{\partial z_k^{(l)}}\right)^2\right]$ is computed using gradient squared expectation over mini-batches. This approximation avoids explicit Hessian computation while capturing curvature information.

Considering ReLU units exhibit different behaviors across samples: activated or inhibited, and removing a ReLU affects the network output only when the corresponding pre-activation satisfies $\mathbf{z}_k^{(l)} \le 0$ (Agarap 2018). Calculating importance based solely on the current sample cannot effectively represent long-term importance.

To address this, we employ a cross-batch accumulation approach that captures activation behavior over multiple batches. We maintain a sliding window of negative-activation probability maps $H^{(l)} = \{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \ldots, \mathbf{h}_M^{(l)}\}$, where each $\mathbf{h}_i^{(l)} \in R^{C\times H\times W}$ stores historical statistics from previous batches. The aggregated negative mask is computed as $A_{\text{neg}}^{(l)} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{h}_i^{(l)}$.

For stable estimation across training batches, we apply Exponential Moving Average (EMA) smoothing with momentum $\lambda = 0.9$. The final importance score for the k-th unit in layer l is then computed as:

$$I_{\text{unit}}^{(l)} = \left(I_{\text{first}}^{(l)} + I_{\text{second}}^{(l)}\right) \odot A_{\text{neg}}^{(l)}. \tag{8}$$

During pruning stage, ReLU units with minimal impact on the loss are prioritized for removal.

Layer Importance Evaluation

Previous evaluation method independently assesses the ReLU importance in each activation unit. However, features at different layers play distinct roles in neural networks (Jha et al. 2021; Cho et al. 2022b). To account for the hierarchical nature of deep networks, we introduce a layer importance

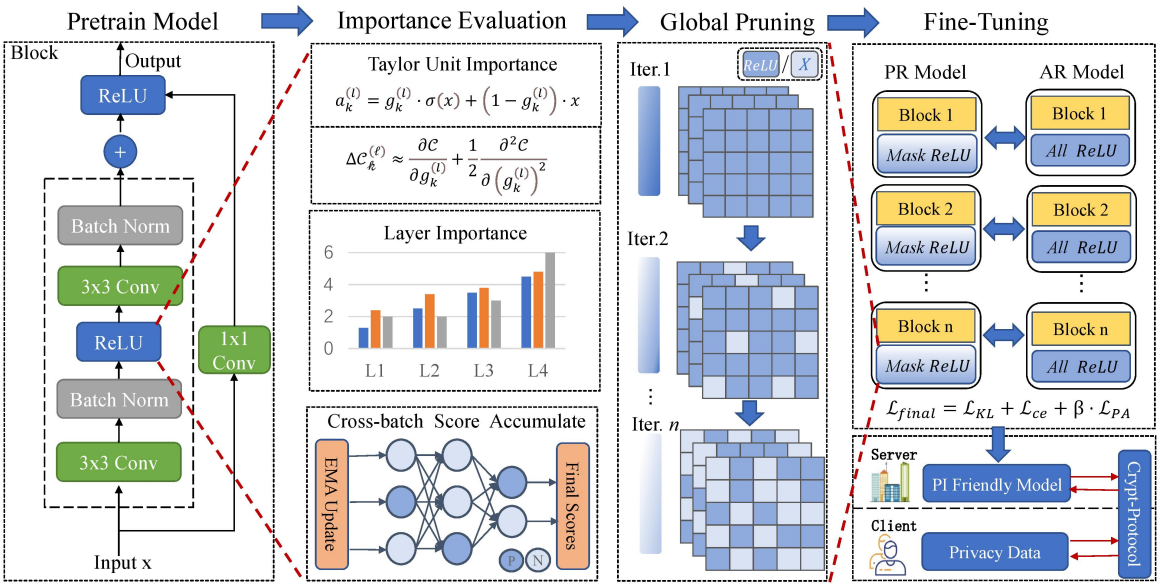

Figure 2: Overview of ReLUPruner framework:ReLUPruner assesses the importance of ReLU units through Taylor importance estimation and layer analysis, uses cumulative mechanism to balance the influence of negative units, adopts a progressive pruning strategy to reserve key units, optimizes the model in the fine-tuning stage, and finally outputs PI-friendly model.

metric that integrates both first-order and second-order gradient information. Our approach computes layer importance as the product of the average gradient L2 norm and the root mean square (RMS) of gradient magnitudes. Specifically, for layer l, the importance score is formulated as:

$$I_{\text{layer}}^{(l)} = \left(\frac{1}{B} \sum_{b=1}^{B} \left\| \nabla \mathbf{a}_b^{(l)} \right\|_2 \right) \cdot \sqrt{\frac{1}{B} \sum_{b=1}^{B} \left(\nabla \mathbf{a}_b^{(l)} \right)^2 + \epsilon} \quad (9)$$

This design balances dimensional consistency and numerical stability. The first factor represents the average gradient magnitude (first moment), while the second factor approximates the square root of the Fisher trace, interpreted as the RMS of gradient norms. The RMS term captures the second moment of gradient distributions, providing a stable estimate of curvature information. By combining both mean and RMS gradient measures, our formulation comprehensively captures the layer's sensitivity to loss changes while maintaining robust numerical behavior across different network architectures and training stages.

Global Pruning and Fine-tuning Strategy

Given the global importance scores $I_{\text{global}}^{(l)} = I_{\text{unit}}^{(l)} \times I_{\text{layer}}^{(l)}$, we implement a global selection process where ReLU units across all layers are ranked by their normalized importance scores. However, abrupt pruning to the target ratio can significantly degrade network performance by disrupting feature representation continuity (Hoefler et al. 2021).

To mitigate this issue, we design a progressive pruning strategy that allows the network to gradually adapt to the evolving architecture. The approach maintains full retention ($\rho = 1$) during initial training stages to preserve learning capacity, then accelerates pruning to approach target sparsity, and finally reduces mask update frequency to stabilize the fine-tuning process.

The complete pruning procedure is outlined in Algorithm 1. During the pruning phase, we employ a composite loss function that integrates both classification and knowledge distillation:

$$\mathcal{L}_{\text{prune}} = \mathcal{L}_{ce} + \mathcal{L}_{kl} \quad (10)$$

where $\mathcal{L}_{ce}$ denotes the cross-entropy loss, $\mathcal{L}_{kl}$ represents the KL divergence knowledge distillation loss from the original teacher model.

Upon completing the pruning stage, we conduct fine-tuning to repair feature representation damage caused by ReLU removal. By incorporating the Post Activation Alignment Loss (PRAM) proposed by SENet (Cho et al. 2022b), we formulate the fine-tuning loss function as:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{ce} + \mathcal{L}_{kl} + \beta \cdot \mathcal{L}_{\text{PRAM}} \quad (11)$$

Experiments

Models and Datasets: To validate the effectiveness of the proposed ReLUPruner and enable comparable analysis with prior studies, comprehensive experiments are conducted on three benchmark datasets: CIFAR-10 (Krizhevsky,

Algorithm 1: Global Progressive ReLU Pruning

Require: Network f_w, dataset $\mathcal{D}$, target retention ratio ρ^*
Ensure: Pruned network f_w^m

Initialize mask $\mathbf{m} \leftarrow \mathbf{1}$, Importance calculator *IC*, scheduler *GS*
for $t = 1$ to T **do**
  GS.set_epoch(t)
  for $(x, y) \in \mathcal{D}$ **do**
    Forward pass: $\mathbf{z}^{(l)}, \mathbf{a}^{(l)} \leftarrow f_w(x, \mathbf{m})$
    Compute loss: $L_{\text{prune}} = \gamma \mathcal{L}_{ce} + \alpha \mathcal{L}_{kl}$
    Backward pass: $\nabla_w \mathcal{L}, \nabla_{\mathbf{a}^{(l)}} \mathcal{L} \leftarrow \text{backward}(\mathcal{L})$
    Update importance:
      $I_{unit}^{(l)} \leftarrow \text{TaylorExpansion}(\mathbf{z}^{(l)}, \nabla_{\mathbf{a}^{(l)}} \mathcal{L})$
      $I_{layer}^{(l)} \leftarrow \text{LayerImportance}(\nabla_{\mathbf{a}^{(l)}} \mathcal{L}, l)$
      $I_{global}^{(l)} \leftarrow I_{unit}^{(l)} \times I_{layer}^{(l)}$
    if *GS*.should_update() **then**
      $\rho \leftarrow$ *GS*.get_ratio() {Adaptive scheduling}
      $\mathbf{m} \leftarrow$ *IC*.generate_mask(ρ)
      GS.update_stats($\mathbf{m}$)
    end if
    Update parameters: $w \leftarrow$ *opt*.step($\nabla_w \mathcal{L}$)
  end for
end for

	Methods	ReLUs(K)	Acc(%)	Latency(s)	Acc/ReLU
ReLU ≤ 100K	Cryptonas	100	92.18	2.26	0.92
	DeepReduce	37	88.9	0.95	2.4
	SNL+	60	92.63	1.28	1.544
	SNL+	12.9	88.23	0.29	6.84
	SENet+	82	93.05	1.81	1.14
	SENet+	49.1	93.6	1.1	1.91
	ours+	50	94.02	0.84	1.881
	ours+	12.9	90.12	0.43	6.986
	ours+	6	87.82	0.12	14.63
ReLU > 100K	Cryptonas	334	94.24	6.14	0.282
	DeepReduce	126	92.5	2.66	0.73
	SNL*	300	95.06	4.3	0.317
	SNL*	240	94.24	3.04	0.392
	SENet+	150	94.91	3.23	0.63
	ours+	150	95.07	1.19	0.634
	ours+	250	95.04	1.28	0.381
	ours+	300	95.18	2.14	0.317

+: starts with ResNet-18 *: starts with WideResNet-22-8

Table 1: Experimental results on CIFAR-10 dataset.

Hinton et al. 2009), CIFAR-100, Tiny-ImageNet (Deng et al. 2009). Three distinct models and their variants are adopted: ResNet-18, ResNet-34 (He et al. 2016), WRN22-8 (Zagoruyko and Komodakis 2016). All implementations are based on PyTorch and executed on NVIDIA RTX 4090 GPUs.

Baseline Models: For baseline models retaining all ReLUs, following established protocols, we train them for 240, 120, and 60 epochs on CIFAR-10/100, Tiny-ImageNet respectively. The initial learning rate (LR) is set to 0.05, decayed by a factor of 0.1 at 62.5%, 75%, and 87.5% of the training epochs. After training completion, we quantify the original ReLU counts and measure inference latency using Delphi (Mishra et al. 2020).

	Methods	ReLUs(K)	Acc(%)	Latency(s)	Acc/ReLU
ReLU ≤ 100K	Cryptonas	100	68.5	2.3	0.685
	DeepReduce	12.3	64.97	0.45	5.282
	SNL+	12.9	66.53	0.291	5.517
	SNL+	49.9	73.75	1.066	1.478
	SENet+	24.6	70.59	0.61	2.87
	SENet+	49.6	75.28	1.13	1.52
	AutoReP+	12.9	67.73	0.15	5.251
	AutoReP+	50	74.28	0.241	1.485
	ours+	12.9	68.28	0.37	5.293
	ours+	24.9	71.98	0.62	2.891
	ours+	50	77.9	0.93	1.558
ReLU > 100K	Cryptonas	334	76	7.5	0.227
	DeepReduce	229.4	76.22	4.61	0.332
	SNL+	120	76.35	2.802	0.636
	SNL+	180	77.65	4.054	0.431
	SENet*	150	78.32	2.62	0.522
	AutoReP*	120	76.32	0.563	0.636
	AutoReP*	150	77.38	0.598	0.516
	AutoReP*	180	77.49	0.636	0.431
	ours*	150	78.61	1.14	0.524
	ours*	180	79.54	1.25	0.441
	ours*	200	79.63	1.42	0.398

+: starts with ResNet-18 *: starts with WideResNet-22-8

Table 2: Experimental results on CIFAR-100 dataset.

Parameter Setting: For the predefined target ReLU retention ratio, we applied ReLUPruner to perform mask training on the three datasets for 150, 100, and 30 epochs, respectively. The initial learning rate was set to 0.1 with a momentum of 0.9. The distillation temperature was fixed at 4.0, and the fine-tuning parameter β in the third stage was configured as 1000.

Experimental Analysis

Pareto Frontier and Cross-Work Comparison We compared ReLUPruner with current SOTA under different ReLU budgets, including SENet (Cho et al. 2022b), AutoReP (Peng et al. 2023), SNL (Kundu et al. 2023), CryptoNAS (Ghodsi et al. 2020), DeepReduce (Jha et al. 2021),Sphynx (Cho et al. 2022a) and presented the results in Figure 3, which report the performance metrics across the three datasets.

We first conducted experiments on the CIFAR-10 dataset, with results shown in Table 1. When the ReLU budget was compressed to 6K (only 7.3% of SENet's budget), ReLUPruner still maintained a baseline accuracy of 87.82%, achieving results close to SNL with half the ReLU budget. At a budget of 50K, ReLUPruner is the method closest to the baseline model's accuracy of 95.55% (94.02%), outper-

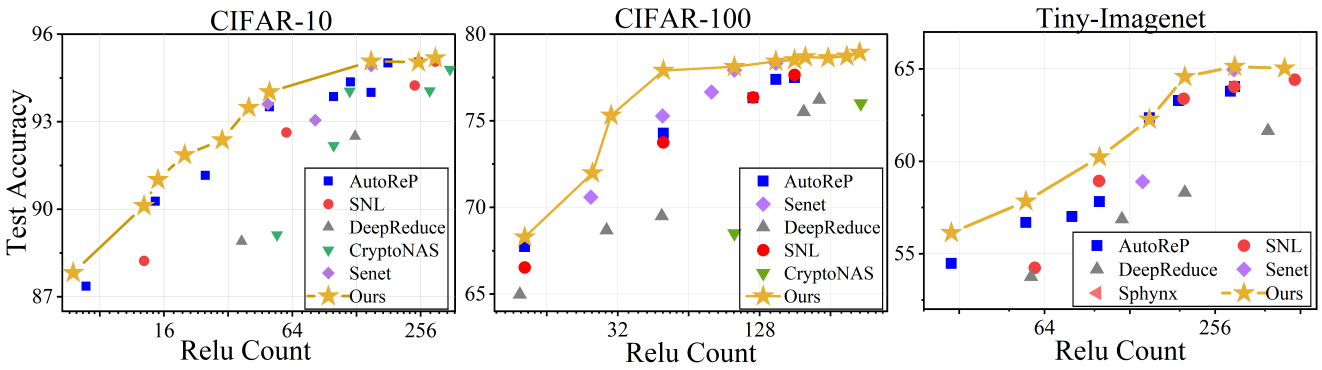

Figure 3: In experiment on the CIFAR-10, CIFAR-100 and Tiny-ImageNet datasets, ReLUPruner has consistently demonstrated an advantage in balancing ReLU counts and test accuracy, and outperforms existing method across all ReLU count ranges. Including AutoReP, SENet, SNL, DeepReDuce, and CryptoNAS.

	Methods	ReLUs(K)	Acc(%)	Latency(s)	Acc/ReLU
ReLU $\leq$ 100K	DeepReduce	57.35	53.75	1.85	0.913
	SNL+	59.4	54.24	1.265	0.918
	SNL+	99.6	58.94	2.12	0.592
	AutoReP+	30	53.47	0.317	1.782
	AutoReP+	55	56.69	0.362	1.031
	ours+	30	56.13	0.421	1.871
	ours+	55	56.84	0.517	1.033
	ours+	100	59.23	0.629	0.592
ReLU $>$ 100K	Sphynx	614.4	60.76	12.5	0.099
	DeepReduce	393	61.65	7.77	0.157
	SNL+	198.1	63.39	4.18	0.32
	SNL+	298.8	64.04	6.281	0.215
	SNL*	488.8	64.42	10.281	0.132
	SENet+	142	58.9	3.08	0.415
	SENet+	298	64.96	6.34	0.218
	AutoReP+	190	63.29	0.562	0.333
	AutoReP+	290	63.79	0.723	0.219
	AutoReP*	300	64.04	1.076	0.213
	ours+	150	61.27	2.48	0.408
	ours+	200	64.58	3.57	0.322
	ours+	300	65.13	4.69	0.217

+: starts with ResNet-18 *: starts with WideResNet-22-8

Table 3: Experimental results on Tiny-ImageNet dataset.

forming Cryptonas' results at a 100K budget. When further increasing the ReLU budget to 300K, ReLUPruner's accuracy surpasses existing solutions, while inference latency is reduced to 2.14 seconds compared to SNL.

Experiments on the CIFAR-100 dataset demonstrate the effectiveness of ReLUPruner. As shown in Table 2, under the low budget condition of 12.9K, ReLUPruner outperforms AutoReP (67.73%) with an accuracy of 68.28%, while the Acc/ReLU metric reaches 5.293. When the budget is increased to 50K, ReLUPruner achieves an accuracy rate of 77.9%, outperforming comparable solutions (AutoReP: 74.28%; SENet: 75.28%), with improvement margins of 3.62% and 2.62%, respectively. Even when reducing the ReLU in the network to 30% of the original (150K), ReLUPruner maintains the network's original accuracy rate, achieving 78.61%, which is similar to the baseline model (79.8%) and outperforms existing solutions (AutoReP: 77.38%), matching SENet's performance. When the budget reaches 200k, compared to DeepReduce at a 229k budget, the accuracy improves by 3.41%.

We further conducted experiments on the more complex Tiny-ImageNet dataset, with results shown in Table 3. Among the methods for optimizing WideResNet at a 300k budget, only ReLUPruner achieved an accuracy of 65.13%, improving by 1.09% over AutoReP (64.04%) and by 0.17% over SENet (64.96%). Under a medium budget of 150K, ReLUPruner utilized the additional 8K budget (61.27%) to achieve a 2.37% accuracy improvement over SENet (58.9%), reaching 0.408 Acc/ReLU, and achieved the levels of Sphynx and DeepReduce at 614K and 393K, respectively, using only half the budget. When the budget was further compressed to 55k and 33k, ReLUPruner achieved accuracy rates of 56.84% and 56.13%, respectively, also outperforming existing solutions.

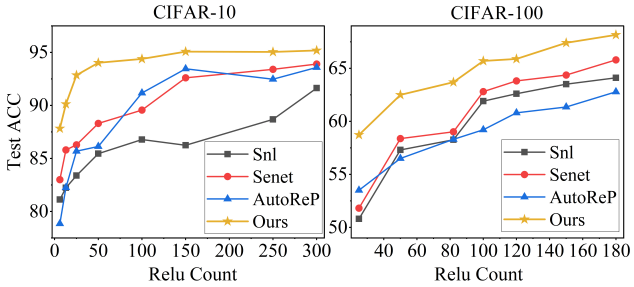

Figure 4: Comparison of pre-fine-tuning test accuracy versus ReLU count on CIFAR-10/100.

Method Analysis

Many previous works AutoReP (Peng et al. 2023), SENet (Cho et al. 2022b), SNL (Kundu et al. 2023) restored the

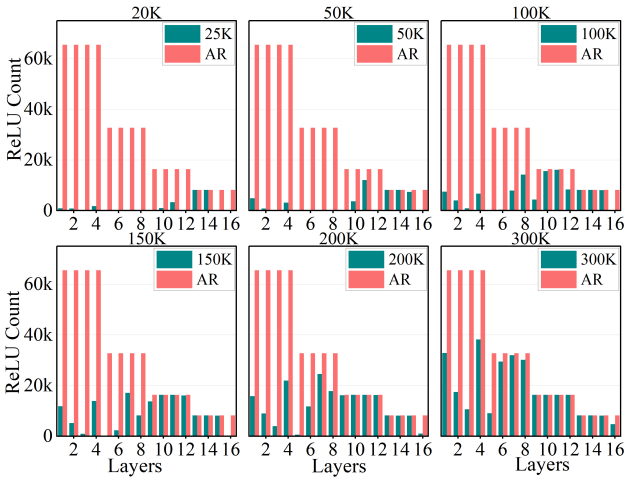

Figure 5: Layer-wise ReLU retention in ResNet-18 on CIFAR-100 under varying budgets (20K–300K).

model's expressive ability in the additional fine-tuning stage after pruning. To independently test the effectiveness of the pruning method without being affected by the results of fine-tuning, we conducted experiments on ResNet-18 and compared the experimental results before fine-tuning. As shown in Figure 4, on the CIFAR-10 dataset, ReLUPruner achieved 87.82% of the results within a 6k budget and continued to lead similar solutions as the budget increased. This result was once again demonstrated on the CIFAR-100 dataset. Before considering fine-tuning methods, ReLUPruner can achieve the highest accuracy rate within the budget constraint, outperforming existing solutions.

We further explored the details of the modifications made by ReLUPruner to the network at different layers. Figure 5 shows the pruning extent of the ResNet-18 network on the CIFAR-100 dataset compared to the original AR model under budgets ranging from 20K to 300K. Compared to the full ReLU model at 100K, ReLUPruner has removed most of the less important ReLU units, retaining them only in key layers. After pruning, the distribution of remaining ReLU units follows the same pattern: later layers (13–15) retain nearly all ReLU functions, while initial layers (1–7) bear the brunt of the reduction. The retention rate of ReLU units does not exceed 1%. This trend aligns with previous research (Jha et al. 2021; Cho et al. 2022b), indicating that ReLU units at different layers contribute differently to the network's importance, and those in deeper layers are more likely to be retained.

Ablation Study

Experiments on the CIFAR-100 dataset (Table 4) show that second-order Taylor expansion significantly improves the performance of ReLU pruning models compared to first-order approximation. When ResNet-18 retains 50k ReLU units, second-order expansion increases the accuracy from 73.32% to 77.90%; and with a budget of 100k ReLU units, the accuracy rate increased from 74.38% to 78.12%. This phenomenon also holds true for WideResNet-22-8: with 120k ReLU units, the accuracy rate increased by 1.24%

Model	#ReLU (k)	Taylor Order	Accuracy (%)
ResNet18	50k	First	73.32
		Second	77.90
ResNet18	100k	First	74.38
		Second	78.12
WideResNet-22-8	120k	First	76.12
		Second	77.36
WideResNet-22-8	150k	First	76.89
		Second	78.61

Table 4: Model Performance in different orders of Taylor expansions

Model	#ReLU (k)	Layer Weight	Accuracy (%)
ResNet18	50k	Disabled	76.13
		Enabled	77.90
ResNet18	100k	Disabled	77.26
		Enabled	78.12

Table 5: Model Performance with Layer-wise Importance

(from 76.12% to 77.36%); and a 1.72% increase at 150k ReLU. Experimental results demonstrate that second-order Taylor expansion can more accurately capture the nonlinear interactions of ReLU units, with particularly significant effects in highly sparse scenarios.

As shown in Table 5, the layer importance mechanism significantly improves model performance under strict ReLU constraints. For ResNet-18 with a 50k ReLU budget, enabling this mechanism achieves 77.90% accuracy compared to 79.13% without it. When the budget increases to 100k ReLUs, the accuracy drops slightly to 77.26% with the mechanism enabled versus 78.12% without it. These results demonstrate that the layer importance strategy preserves deep semantic features by preventing uniform pruning from disrupting hierarchical representations. Further analysis reveals that tighter ReLU budgets amplify the performance benefits of this mechanism.

Conclusion

We presents ReLUPruner, a novel framework for efficient private inference via progressive ReLU pruning guided by Taylor importance estimation. By deriving a second-order Taylor expansion of the loss function with respect to ReLU gating variables, our method explicitly quantifies the impact of individual ReLU units on model expressiveness, enabling precise identification of redundant activations. Extensive experiments demonstrate that ReLUPruner achieves state-of-the-art accuracy-ReLU trade-offs, offering a principled, efficient solution to accelerate private inference while minimizing accuracy degradation.

References

Agarap, A. F. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Brutzkus, A.; Gilad-Bachrach, R.; and Elisha, O. 2019. Low Latency Privacy Preserving Inference. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 812–821. PMLR.

Cheng, H.; Zhang, M.; and Shi, J. Q. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cheon, J. H.; Kim, A.; Kim, M.; and Song, Y. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, 409–437. Springer.

Cho, M.; Ghodsi, Z.; Reagen, B.; Garg, S.; and Hegde, C. 2022a. Sphynx: A Deep Neural Network Design for Private Inference. *IEEE Secur. Priv.*, 20(5): 22–34.

Cho, M.; Joshi, A.; Reagen, B.; Garg, S.; and Hegde, C. 2022b. Selective Network Linearization for Efficient Private Inference. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 3947–3961. PMLR.

Couteau, G.; Devadas, L.; Hegde, A.; Jain, A.; and Servan-Schreiber, S. 2025. Multi-Key Homomorphic Secret Sharing. In Fehr, S.; and Fouque, P., eds., *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part V*, volume 15605 of *Lecture Notes in Computer Science*, 3–33. Springer.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Ghodsi, Z.; Veldanda, A. K.; Reagen, B.; and Garg, S. 2020. CryptoNAS: Private Inference on a ReLU Budget. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K. E.; Naehrig, M.; and Wernsing, J. 2016. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In Balcan, M.; and Weinberger, K. Q., eds., *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, 201–210. JMLR.org.

Hao, M.; Li, H.; Chen, H.; Xing, P.; Xu, G.; and Zhang, T. 2022. Iron: Private Inference on Transformers. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual ConferenceB on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hoefler, T.; Alistarh, D.; Ben-Nun, T.; Dryden, N.; and Peste, A. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241): 1–124.

Hu, P.; Sun, L.; Hu, C.; Dai, L.; Guo, S.; and Yu, M. 2024. DReP: Deep ReLU pruning for fast private inference. *Journal of Systems Architecture*, 152: 103156.

Jha, N. K.; Ghodsi, Z.; Garg, S.; and Reagen, B. 2021. DeepReDuce: ReLU Reduction for Fast Private Inference. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, 4839–4849. PMLR.

Juvekar, C.; Vaikuntanathan, V.; and Chandrakasan, A. P. 2018. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In Enck, W.; and Felt, A. P., eds., *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, 1651–1669. USENIX Association.

Kilian, J. 1988. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 20–31.

Knott, B.; Venkataraman, S.; Hannun, A. Y.; Sengupta, S.; Ibrahim, M.; and van der Maaten, L. 2021. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 4961–4973.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kundu, S.; Lu, S.; Zhang, Y.; Liu, J. T.; and Beerel, P. A. 2023. Learning to Linearize Deep Neural Networks for Secure and Efficient Private Inference. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Liu, J.; Juuti, M.; Lu, Y.; and Asokan, N. 2017. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 619–631.

Lou, Q.; Shen, Y.; Jin, H.; and Jiang, L. 2020. Safenet: A secure, accurate and fast neural network inference. In *International Conference on Learning Representations*.

Mishra, P.; Lehmkuhl, R.; Srinivasan, A.; Zheng, W.; and Popa, R. A. 2020. Delphi: A Cryptographic Inference Service for Neural Networks. In Capkun, S.; and Roesner, F.,

eds., *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, 2505–2522. USENIX Association.

Mohassel, P.; and Zhang, Y. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, 19–38. IEEE Computer Society.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11264–11272.

Pang, Q.; Zhu, J.; Möllering, H.; Zheng, W.; and Schneider, T. 2024. BOLT: Privacy-Preserving, Accurate and Efficient Inference for Transformers. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, 4753–4771. IEEE.

Peng, H.; Huang, S.; Zhou, T.; Luo, Y.; Wang, C.; Wang, Z.; Zhao, J.; Xie, X.; Li, A.; Geng, T.; Mahmood, K.; Wen, W.; Xu, X.; and Ding, C. 2023. AutoReP: Automatic ReLU Replacement for Fast Private Network Inference. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, 5155–5165. IEEE.

Wang, J.; Su, W.; Huang, Z.; Chen, J.; Luo, C.; and Li, J. 2024. Practical privacy-preserving MLaaS: when compressive sensing meets generative networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 15502–15510.

Yan, G.; Zhang, Y.; Guo, Z.; Zhao, L.; Chen, X.; Wang, C.; Wang, W.; Meng, D.; and Hou, R. 2025. Comet: Accelerating Private Inference for Large Language Model by Predicting Activation Sparsity. In Blanton, M.; Enck, W.; and Nita-Rotaru, C., eds., *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, 2827–2845. IEEE.

Yao, A. C. 1986. How to Generate and Exchange Secrets (Extended Abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, 162–167. IEEE Computer Society.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.