

Securing Neural Networks with Knapsack Optimization

Yakir Gorski, Amir Jevnisek, Shai Avidan

yakirg320@gmail.com, amirjevn@mail.tau.ac.il, avidan@eng.tau.ac.il

Abstract

MLaaS Service Providers (SPs) holding a Neural Network would like to keep the Neural Network weights secret. On the other hand, users wish to utilize the SPs' Neural Network for inference without revealing their data. Multi-Party Computation (MPC) offers a solution to achieve this. Computations in MPC involves communication, as the parties send data back and forth. Non-linear operations are usually the main bottleneck requiring the bulk of communication bandwidth. In this paper, we focus on ResNets, which serve as the backbone for many Computer Vision tasks, and we aim to reduce their non-linear components, specifically, the number of ReLUs.

Our key insight is that spatially close pixels exhibit correlated ReLU responses. Building on this insight, we replace the per-pixel ReLU operation with a ReLU operation per patch. We term this approach 'Block-ReLU'. Since different layers in a Neural Network correspond to different feature hierarchies, it makes sense to allow patch-size flexibility for the various layers of the Neural Network. We devise an algorithm to choose the optimal set of patch sizes through a novel reduction of the problem to the Knapsack Problem. We demonstrate our approach in the semi-honest secure 3-party setting for four problems: Classifying ImageNet using ResNet50 backbone, classifying CIFAR100 using ResNet18 backbone, Semantic Segmentation of ADE20K using MobileNetV2 backbone, and Semantic Segmentation of Pascal VOC 2012 using ResNet50 backbone. Our approach achieves competitive performance compared to a handful of competitors. Our source code is publicly available: https://github.com/yg320/secure_inference.

Introduction

With the recent surge in the popularity of machine learning algorithms, there is a similar increase in the interest of Private Inference (PI). In a typical PI scenario, two parties, a client with an image and an ML-provider server with a deep-learning model, seek to collaborate for the purpose of running inference. However, privacy concerns often impede such cooperation as both the client's image and the server's model may contain information that neither party is willing to share.

Secure Inference protocols enable Private Inference, yet they come at the expense of increased runtime and communication bandwidth consumption due to the necessary communication rounds for inference evaluation. Non-linear

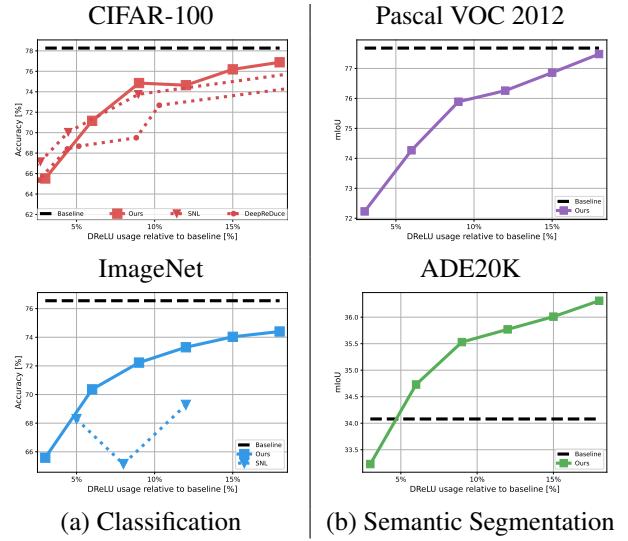


Figure 1: **Task Performance vs DReLU budget:** (a) Classification results for CIFAR-100 and ImageNet. We show competitive results with respect to the alternatives. (b) Semantic Segmentation results. There are no published results for Secure Semantic Segmentation to compare against.

functions particularly contribute to these communication rounds. Neural Networks involve both linear and non-linear operations. For example, ResNet18 classifying CIFAR-100 contains over 500K ReLU activations. ReLUs are the product of the input and a non-linear function which is termed DReLU. DReLU is an indicator function which is set to 1 if the input is positive and is 0 when the input is negative. The goal of this paper is to reduce the number of DReLUs in a Neural Network.

The trade-off between reducing DReLUs in the network and task performance is illustrated in Figure 1. In Figure 1(a), we demonstrate competitive results against current state-of-the-art methods in the context of classifying CIFAR-100 with ResNet18. Furthermore, our approach outperforms the current SOTA method in classifying larger images like ImageNet. Additionally, we present the first evaluation of secure inference for Semantic Segmentation, as depicted in Figure 1(b).

Secure inference aims for privacy-preserving computation in scenarios where a service provider and a client jointly perform calculations without exposing their data. While Homomorphic Encryption (HE) theoretically allows direct work on encrypted data, its real-world speed is impractical. Secure inference typically employs Multi-Party Computation (MPC), involving rounds of communication. However, a central challenge lies in the high computational cost of comparison operations like DReLU, which tends to dominate runtime and bandwidth.

Our main insight lies in the strong correlation of DReLU outcomes among neighboring pixels. To leverage this, we propose a strategy of employing a single ReLU operation per patch. We term ReLU per patch bReLU. However, a crucial question arises: what should be the optimal patch size? To address this we use a novel, Knapsack-based optimization strategy, to find an optimal configuration of patch sizes for all layers in the network.

Our main objective is to minimize DReLU counts while causing as little degradation to the network’s performance as possible. Intuitively, we view this tradeoff as a variant of the Rate-Distortion tradeoff (Berger 2003). We devise a network distortion measure, serving as a proxy for evaluating the network performance degradation under the replacement of ReLUs with bReLUs. Utilizing this proxy, we formulate our problem as an optimization goal with constraints. Given a DReLU count budget, our aim is to minimize the network’s degradation while adhering to the budget. This problem aligns with the Multiple-Choice Knapsack Problem (Kellerer et al. 2004), and we implement a pseudo-polynomial time Dynamic Programming solution that allows us to find the optimal set of patch sizes based on our assumptions.

Additionally, we implement bit truncation (Nishide and Ohta 2007; Ghodsi et al. 2021) to lower the cost of comparison operations. We achieve this by disregarding some of the most and least significant bits of the activation layers. To summarize, our method comprises four key steps: (1) quantifying the distortion of the Neural Network for each layer for a set of patch-size hypotheses, (2) solving for the Block-ReLU patch sizes using a Knapsack-based optimization strategy, (3) replacing ReLUs with Block-ReLUs to reduce non-linear operations in the network and fine-tune the network, and finally (4) employing bit truncation to further reduce computations and achieve a speedup.

Existing benchmarks for Secure Inference primarily focus on small images like CIFAR-100’s 32×32 resolution or Tiny-ImageNet’s 64×64 , mostly limited to Classification tasks. To extend these benchmarks, we evaluate Private Inference on large images like ImageNet, which includes images with an average resolution of 469×387 , and we are the *first* to evaluate on Semantic Segmentation benchmarks like ADE20K and Pascal VOC12. We evaluate our model both on classical CIFAR-100 classification with ResNet18, as well as on ImageNet classification using ResNet50 and Semantic Segmentation of ADE20K using DeepLabV3 with MobileNetV2 backbone, and Semantic Segmentation of Pascal VOC 2012 using DeepLabV3 with ResNet50 backbone.

Since ReLU operations contribute the most to communication bandwidth, we save a considerable amount of network traffic. To validate this, we implement a semi-honest secure 3-party setting of the SecureNN protocol. We use 3 instances within the same AWS EC2 region. ImageNet classification is executed in about 5.5 seconds, and CIFAR100 classification in about 1.5 seconds. ADE20K Semantic Segmentation is executed in about 32 seconds and Pascal VOC 2012 on 85 seconds. Classification and Segmentation performances measured are all comparable to OpenMMLab’s non-secure models (Contributors 2020a,b).

To summarize, our main contributions are:

- We develop a generic, Knapsack-based, data-driven algorithm that greatly reduces the number non-linear operations (DReLUs) in the network.
- We demonstrate a significant runtime reduction with a marginal trade-off in accuracy across classical benchmarks for both low-resolution, as well as in the previously unaddressed large images, and the task of Semantic Segmentation.
- We present a secure semantic segmentation algorithm that preserves the model accuracy while being $\times 10$ faster than a comparable secure baseline protocol.
- We build and release an optimized, a purely Pythonic, wrapper code over OpenMMLab based packages that secures models taken from their model zoo.

Related Work

Privacy Preserving Deep Learning The research on privacy preserving deep learning shows significant differences across various aspects. These include the number of parties involved, threat models (Semi-honest, Malicious), supported layers, techniques used (e.g., Homomorphic encryption, garbled circuits, oblivious transfer, and secret sharing) and the capabilities provided (training and inference).

The pioneers in the field of performing prediction with neural networks on encrypted data were CryptoNets (Gilad-Bachrach et al. 2016). They employed leveled homomorphic encryption, replacing ReLU non-linearities with square activations in order to perform inference on ciphertext. SecureML (Mohassel and Zhang 2017) utilized three distinct sharing methods: Additive, Boolean, and Yao sharing, along with a protocol to facilitate conversions between them. They used linearly homomorphic encryption (LHE) and oblivious transfer (OT) to precompute Beaver’s triplets. MiniONN (Liu et al. 2017) proposed to generate Beaver’s triplets using additively homomorphic encryption together with the single instruction multiple data (SIMD) batch processing technique. GAZELLE (Juvekar, Vaikuntanathan, and Chandrakasan 2018) suggested to use packed additively homomorphic encryption (PAHE) to make linear layers more communication efficient. FALCON (Li et al. 2020) achieved high efficiency by running convolution in the frequency domain, using Fast Fourier Transform (FFT) based ciphertext calculation. Chameleon (Riazi et al. 2018) builds upon ABY (Patra et al. 2021) and employs a Semi-honest Third Party (STP) dealer to generate Beaver’s triplets in an offline phase. SecureNN (Wagh, Gupta, and Chandran 2019)

suggested three-party computation (3PC) protocols for secure evaluation of deep learning components. The Porthos component of CrypTFlow (Kumar et al. 2020) is an improved semi-honest 3-party MPC protocol that builds upon SecureNN. FALCON (Wagh et al. 2020), combines techniques from SecureNN and ABY³ (Mohassel and Rindal 2018), replacing the MSB and Share Convert protocols in SecureNN with cheaper Wrap₃ protocols.

ReLU Savings There are several methods to reduce ReLU expense through non-cryptographic means. One family of such methods, replaces non-linear functions with polynomial approximations for the non-linearities (Hesamifard et al. 2018; Khan, Bakas, and Michalas 2021). Another family of methods leverage Network Architecture Search (NAS) algorithms to augment the network architecture such that the result network reduces ReLU usages. Such works include DELPHI (Srinivasan, Akshayaram, and Ada 2019) for which ReLUs are approximated with quadratic polynomial and a NAS algorithm is employed to automatically discover which ReLUs are replaced with the approximations. Another is CryptoNAS (Ghodsi et al. 2020) in which the authors developed a NAS over a fixed-depth, skip connections architectures to reduce ReLU count.

Two prominent state-of-the-art methods in this domain are DeepReDuce (Jha et al. 2021) and Selective Network Linearization (SNL) (Cho et al. 2022). DeepReDuce is composed of three sequential stages: ReLU “culling,” ReLU “thinning,” and ReLU “reshaping.” However, these stages necessitate several manual design decisions and involve multiple hyper-parameters. In contrast, SNL (Cho et al. 2022) introduces an individual parameter for each ReLU within the Neural Network. We refer to this parameter as the convex combination parameter. It governs whether a given ReLU is activated or replaced with an identity function. SNL employs a gradient-based approach to replace ReLUs with identity operations iteratively, solving for these parameters. During each iteration, the replacement process is coupled with the optimization of neural network parameters. Nevertheless, it’s important to note that the final result network obtained through SNL’s process requires re-training.

While the optimization-based method SNL displays promise by relaxing some of DeepReDuce’s design constraints, our paper addresses the limitations it introduces. Unlike SNL, our approach handles networks that accommodate varying input image shapes, such as those encountered in Semantic Segmentation tasks. In contrast to SNL’s stopping criterion, which often falls short of precisely attaining the desired non-linearity budget, our method consistently achieves an accurate budget. Additionally, SNL’s iterative process necessitates network fine-tuning during each optimization iteration before arriving at a new Neural Network architecture. This considerably slows down their process compared to our inference and Knapsack-based pre-processing. Further elaboration on this topic can be found in the Supplementary Material.

In (Helbitz and Avidan 2021), which is the work that has most influenced us, the authors used statistics from neighboring activations and shared DReLU among them. How-

ever, they did not provide a satisfactory method for determining the neighborhood. Finally, Circa (Ghodsi et al. 2021) proposed the Stochastic ReLU layer and showed that we can use prior knowledge about the absolute size of activations to reduce the complexity of garbled circuits while maintaining a low error probability. They also demonstrated that neural networks can handle clipping of the least significant bits, which further reduces circuit complexity. While we share similar observations with Circa, our approximate ReLU layer differs in its application, as we ignore the shares’ most significant bits instead of the negligible probability event of a shares summation overflow. This enables us to better control bandwidth with error probability, which is especially useful when reducing shares precision, as in (Wagh et al. 2020).

Network Pruning Network pruning is a technique applied to reduce model complexity by removing weights that contribute the least to model accuracy. Our method can be seen as a specialized pruning technique that targets DReLU operations instead of weights. Different approaches use various measures of importance to identify and eliminate the least important weights, such as magnitude-based pruning (e.g., (Han, Mao, and Dally 2015; Han et al. 2015)), similarity and clustering methods (e.g., (Son, Nah, and Lee 2018; RoyChowdhury et al. 2017; Dubey, Chatterjee, and Ahuja 2018)), and sensitivity methods (e.g., (Hassibi and Stork 1992)). Some methods, including ours, score filters based on the reconstruction error of a later layer (e.g., (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017; Kamma and Wada 2019)). The pioneers of Knapsack based network pruning (Aflalo et al. 2020) used the 0/1 Knapsack solution to eliminate filters in a neural network, whereby an item’s weight was based on the number of FLOPs, and its value was based on the first-order Taylor approximation of the change to the loss. Similarly, in (Shen et al. 2022) the authors optimized network latency using the Knapsack paradigm. The authors of (Hubara et al. 2021) applied Integer Programming to optimize post training quantization.

Method

Our approach is built upon the observation that ReLUs within a spatial neighborhood often exhibit similar non-linear activation responses. Leveraging this insight, we introduce Block-ReLU (bReLU), a non-linear activation function defined over a patch of activations. Substituting ReLUs with Block-ReLUs in the network reduces the number of non-linear operations, a desirable trait for Private Inference. However, this replacement introduces a certain degree of error. We aim at reaching a desired ReLU budget while minimizing the error incurred by our replacement.

To accomplish this, we develop a distortion measure that quantifies the degradation resulting from replacing ReLUs with Block-ReLUs of varying patch sizes. Assuming distortion additivity, we iterate ReLU channels in the Neural Network. In each iteration we replace the *current channel*’s ReLU with Block-ReLU. We evaluate the network distortion for different patch-sizes for that replacement. With these distortion values in hand, we formulate the optimal patch sizes

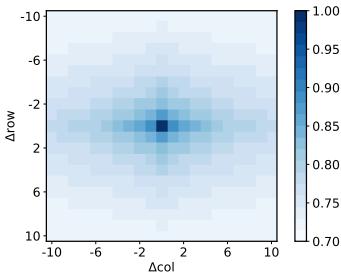


Figure 2: Activation Statistics for MobileNetV2: The probability that two activation units in the same channel have the same sign, based on their spatial distance. The lowest probability is about $0.7 = 70\%$.

as a variant of the Knapsack optimization problem. Solving this optimization yields the optimal patch sizes under our assumptions.

Finally, we fine-tune the neural network and apply the previously suggested bit-truncation (Nishide and Ohta 2007; Ghodsi et al. 2021) for yet another runtime and bandwidth consumption Secure Inference performance boost. For Secure Inference, we follow the SecureNN (Wagh, Gupta, and Chandran 2019) protocol, which is concisely summarized in the appendix.

Approach

Following SecureNN notations, we refer to the ReLU decision of an activation unit as its DReLU. Mainly:

$$\text{DReLU}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Then, ReLU is defined as:

$$\text{ReLU}(x) = x \cdot \text{DReLU}(x) \quad (1)$$

Block-ReLU (bReLU) Our key insight is that neighboring activation units tend to have similar signs. To demonstrate this, we utilize a pre-trained Segmentation network trained on the ADE20K dataset with MobileNetV2 backbone. We evaluate the empirical probability of two activation units residing in the same channel having the same sign. Figure 2 shows the probability map as a function of horizontal and vertical distances. Remarkably, even for lag = 10, the probability remains at approximately $\sim 70\%$. To leverage this insight, we introduce the block ReLU (bReLU) layer.

Block ReLU utilizes the local spatial context of an activation unit to estimate its ReLU decision. Specifically, each activation channel is partitioned into rectangular patches of a specified size. The dimensions of the rectangle will be determined next using a Knapsack optimization solution.

Let x be an activation unit, and $P(x)$ be its neighborhood induced by the rectangle partition. We define the patch ReLU decision (pDReLU) as the DReLU of the mean activation value of the neighborhood:

$$\text{pDReLU}(x) = \text{DReLU}\left(\frac{1}{|P(x)|} \sum_{a \in P(x)} a\right) \quad (2)$$

Input Activation						Output Activation					
-0.5	-0.8	-0.5	0.4	-0.6	0.7	0.0	0.0	0.0	0.4	-0.6	0.7
-0.7	0.2	0.1	0.9	0.9	1.0	0.0	0.0	0.0	0.9	0.9	1.0
0.2	0.8	0.2	-0.5	0.1	-0.2	0.2	0.8	0.2	0.0	0.0	0.0
-0.2	0.9	-0.1	-0.7	-0.9	0.5	-0.2	0.9	-0.1	0.0	0.0	0.0
0.3	-0.2	-0.4	0.6	0.2	-0.8	0.0	0.0	0.0	0.0	0.0	0.0
-0.8	0.5	0.1	-0.1	-0.2	-0.5	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3: Block-ReLU (bReLU): Left: 6×6 channel input, Right: bReLU output with a patch-size of 2×3 . We convert 36 DReLU operations to a mere 6 operations, with the cost of 12 sign flips.

The ReLU decision of each pixel in the neighborhood is then replaced by the patch decision of that pixel.

$$\text{bReLU}(x) = x \cdot \text{pDReLU}(x) \quad (3)$$

As a result, we perform a single DReLU operation in the $P(x)$ patch, instead of $|P(x)|$ individual DReLU operations. This is our major contribution in terms of cutting the number of non-linear operations. The extent in which we save non-linear activations is determined by the patch area (or neighborhood size) $|P(x)|$. Figure 3 illustrates a bReLU with a patch size of 2×3 operating on a 6×6 activation channel.

Knapsack Optimization Our goal is to adhere to some DReLU budget \mathcal{B} , with minimal degradation to the network performance. To achieve this, we solve for the Block-ReLU patch size. We draw inspiration from the Rate-Distortion (Berger 2003) theory. We devise a distortion measure which serves as a proxy for the network performance and the rate is addressed as the ratio between the number of bReLUs to the number of ReLUs: $\frac{\#\text{bReLUs}}{\#\text{ReLU}}$.

Different layers in the neural correspond to different feature hierarchies and different channels within the same layer correspond to different features. Therefore, determining a neighbourhood size or patch-size for the bReLU operation is *channel* dependent. We enumerate the channels across the network and denote C_i the i^{th} channel in a neural network. For example, in a network consisting of two layers, with 32 channels in the first layer and 64 in the second layer, C_{40} would refer to the 8th channel in the second layer.

Next, we define P_i as the list of patch-sizes available for selection by C_i , and P_{ij} as the j -th item within this list. $\mathcal{D}(i, j)$ is then defined as the distortion caused to the network as a result of replacing the ReLUs of C_i with a bReLU of patch size P_{ij} .

The parameter we optimize is the patch-size P_{ij} . Evaluating the network’s performance for each patch-size configuration is exponentially complex with the number of patches (proportional to L^k , where L is the number of channels in the network and k is the number of patch-size options).

To address this challenge, we introduce two relaxations for this task. First, we devise a distortion measure to serve as a proxy for the network performance degradation. We choose to measure this distortion by quantifying the change

in the network activation for the same inputs. This makes distortion a general purpose solution for both classification and Semantic Segmentation. Clearly, future work can discuss other types of distortion methods.

Formally, let F be a neural network, and $F^{i,j}$ denote the network obtained by replacing the ReLUs of C_i of F with bReLU of a patch size P_{ij} . Furthermore, let $F(X)$ and $F^{i,j}(X)$ be the last activation layers of F and $F^{i,j}$ operating on an image X . Then the distortion $\mathcal{D}(i, j)$ is defined as follows:

$$\mathcal{D}(i, j) = \mathbb{E}_X [\|F^{i,j}(X) - F(X)\|^2] \quad (4)$$

The distortion is calculated as the average L_2 distance between the last activation layers of the original pre-trained network and a similar network, which has the same weights but replaces the ReLU in C_i with a bReLU having a patch-size of P_{ij} . This measure is evaluated across a subset of train images.

The second relaxation that we make is to approximate the total distortion resulting from replacing ReLUs with bRE-LUs for a number of channels as the additive distortion. This allows us to cast the optimal patch-shape as a Knapsack optimization problem. The cost for optimization problem is determined by the number of DReLU remaining in C_i of $F^{i,j}$, which we denote as $\mathcal{W}(i, j)$. It is important to note that $\mathcal{W}(i, j)$ is influenced by both the patch-size and the activation channel size. Additionally, we define m as the number of channels in F , and S_i as the set of indices $1, \dots, |P_i|$. With these notations, our optimization problem boils down to the Multiple-Choice Knapsack Problem (Kellerer et al. 2004):

$$\begin{aligned} \min_{\varphi_{ij}} \quad & \sum_{i=1}^m \sum_{j \in S_i} \mathcal{D}(i, j) \cdot \varphi_{ij} \\ \text{subject to} \quad & \sum_{i=1}^m \sum_{j \in S_i} \mathcal{W}(i, j) \cdot \varphi_{ij} \leq \mathcal{B}, \\ & \sum_{j \in S_i} \varphi_{ij} = 1, \quad i = 1, \dots, m, \\ & \varphi_{ij} \in \{0, 1\}, \quad i = 1, \dots, m, \quad j \in S_i \end{aligned} \quad (5)$$

The indicator variable $\varphi_{ij} = 1$ indicates that the P_{ij} patch size has been selected for C_i . The second constraint ensures that only a single patch size can be selected for this channel, thereby preventing the selection of multiple patch sizes.

We use Dynamic Programming to solve this optimization problem. We iteratively populate a table DP consisting of m rows and \mathcal{B} columns, using the following recursive formula:

$$DP[i, j] = \min_{1 \leq l \leq S_i} \{DP[i - 1, j - \mathcal{W}(i, l)] + \mathcal{D}(i, l)\} \quad (6)$$

As this table is column-independent, we can parallelize the process of finding the solution $DP[m, \mathcal{B}]$. To fully leverage this property, we implement a GPU version of the algorithm. The code for the parallelized solver will be made publicly available upon acceptance.

Analysis Our optimization cost in Equation 5 assumes that the total distortion replacing ReLUs with bRE-LUs across multiple channels is additive. This implies that the distortion caused by the ReLU-bReLU replacement in two channels equals to the sum of distortions caused by the replacement in each channel individually. Notably, this assumption can be relaxed, with a sufficient condition being a monotonically non-decreasing relationship between the real distortion and the additive distortion. Further exploration of this relationship is detailed in the supplementary material.

Homogeneous Channels Some channels tend to have uniform signs across the entire channel, where all activations are either positive or negative. To leverage this observation, we extended the P_i lists, which hold the different options for patch-sizes, by adding an additional special item: the identity channel. This identity channel has a weight of $\mathcal{W} = 0$, indicating that no DRE-LUs are employed in this case.

Secure Inference Considerations

Approximate DReLU In protocols that utilize additive sharing, the cost of a comparison operation is typically proportional to the number of bits used to represent the shares. This relationship is evident in the depth of a Yao’s Garbled Circuit, and in SecureNN-based protocols. Previous observations (Nishide and Ohta 2007; Ghodsi et al. 2021) indicate that by allowing for a small probability of DReLU error, the complexity of comparison operations can be substantially reduced. We analyze the probability of a DReLU error as a function of the number of most and least significant bits ignored. We use ResNet50 with bReLU layers trained on the ImageNet dataset by and examine millions of activation values. Based on this analysis, we choose to disregard 43 of the most significant bits and 5 of the least significant bits, resulting in a DReLU error probability of about 5×10^{-4} .

MaxPool and ReLU6 MaxPool and ReLU6 are computationally costly layers for secure inference. We suggest to substitute ResNet’s MaxPool layer with an AveragePool layer and MobileNet’s ReLU6 layers with ReLU layers. If required, we also adjust the models through fine-tuning. Exact fine-tuning details are summarized in the supplemental.

Security Analysis

Our method offers the same level of security as the underlying cryptographic protocol, SecureNN in our case. Accordingly, we protect both the client image and server model weights under the relevant thread model. However, we do make the patch sizes public, which typically consists of approximately 40K discrete parameters that we infer based on the training data statistics. It is important to note that any hyperparameters or architectural structures that are revealed may result in some degree of training data information leakage. Therefore, researchers should be aware of this gray area and determine what level of model disclosure is acceptable. Other than image size, the user’s image information is entirely protected.

Experiments

Implementation Details To evaluate the efficacy of our method, we use the standard Classification benchmark classifying CIFAR-100 (Krizhevsky, Nair, and Hinton 2010) with the ResNet18 backbone. We also evaluate the classification of ImageNet (Deng et al. 2009), using ResNet50 (He et al. 2016) backbone and perform an evaluation on two Semantic Segmentation datasets: ADE20K (Zhou et al. 2017) using DeepLabV3 (Chen et al. 2017) with MobileNetV2 (Sandler et al. 2018) backbone and Pascal VOC 2012 (Everingham et al. 2010) using DeepLabV3 and ResNet50 backbone. Pre-trained networks are obtained from the OpenMMLab model zoo, other than the missing CIFAR100 models which were trained from scratch.

We evaluate distortion through forward passes in the Neural Network. The distortions are evaluated for the different patch-sizes and channels. The optimal patch-sizes were determined using our CUDA-based Multiple-Choice-Knapsack solver. ReLU layers were replaced with bReLU layers, parameterized by the Knapsack-optimal patch-sizes, and the models were fine-tuned. Details of the Knapsack search parameters (e.g. patch-sizes lists), fine-tuning hyper parameters and computing infrastructure are provided in the supplementary material.

Evaluation We perform two kinds of evaluations: (1) we measure the task performance of our secure network and compare it to DeepReDuce and the SOTA SNL. We also present the baseline non-secure task performance of the Neural Network on the original task (Classification or Semantic Segmentation), (2) we evaluate runtime and bandwidth consumption of our network under SecureNN, a specific Secure Inference protocol.

Figure 1 shows the tradeoff between ReLU savings and task performance. We evaluate our method against DeepReDuce and SNL for the classification of CIFAR-100 and against SNL for ImageNet. We use black dashed line to indicate the non-secure baseline task performance. For almost all classification benchmarks, we outperform competitors. It is important to highlight that SNL in its current form, cannot be used on variable size images and is therefore not evaluated for the Semantic Segmentation benchmark.

The second type of evaluation we conduct is a Secure Inference protocol test which measures the communication bandwidth and latency (runtime) of a secure protocol using our suggested secure network. For this, we set up 3 Amazon EC2 c4.8xlarge Ubuntu-running instances in the same region (eu-west-1). We developed our own Numba (Lam, Pitrou, and Seibert 2015) based Python implementation of the SecureNN protocol. Further secure inference implementation details are in the supplementary material. Runtime performance is evaluated for 3 images for Semantic Segmentation and 10 images for classification tasks. Baseline models are downloaded from the OpenMMLab model zoo and include neither bReLU nor approximate ReLU components.

To assess the effectiveness of our network in terms of Secure Inference, we evaluate how it reduces runtime and bandwidth consumption compared to a baseline model con-

	Classification		Segmentation	
	ImageNet ResNet50	CIFAR-100 ResNet18	ADE20K MobileNetV2	VOC12 ResNet50
Non-secure	70.36	70.90	34.73	74.17
Ours (Secure)	70.27	71.16	34.64	74.21

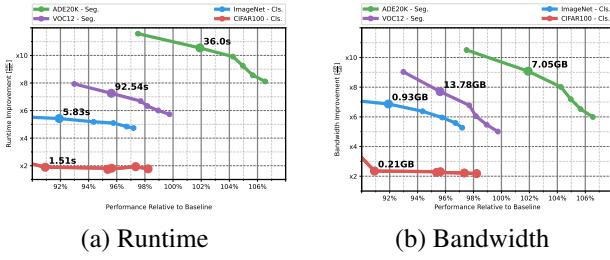
Table 1: **Secure vs. Non-secure Task Performance** With a DReLU budget of 6%, the task performance, measured by mIoU for Semantic Segmentation and Accuracy for Classification, remains largely unchanged switching from Non-Secure to our (Secure) network.

taining all ReLUs and utilizing full ReLU resolution. While Figure 1 shows the performance of the network on the task vs. the DReLU budget, Figures 4(a)+(b) shows the runtime and communication bandwidth savings of the entire protocol compared to the non-secure version. Figures 4(a), (b) illustrate the runtime and bandwidth savings achieved by our network in comparison to the baseline model. Each point on the graph corresponds to a distinct operating point in terms of task performance, with mIoU for Semantic Segmentation and Accuracy for Classification. Lower task performance corresponds to a reduced DReLU budget. Higher improvement correspond to better runtime or bandwidth communication consumption performance over the baseline non secure model. The graph reflects the tradeoff between task performance and runtime and bandwidth consumption. For completeness, we compare secure and non secure performance in Table 1, under a fixed DReLU budget of 6%.

Secure Inference Analysis Table 2 displays the cost of communication for the various layers in the SecureNN protocol, with the CrypTFlow convolution optimization being utilized. We define h as the activation dimension. i and o are the number of activation input and output channels. f is the convolution kernel size. ℓ is the number of bits used to represent activation values. ℓ^* is the number of bits used in approx. DReLU. P is a list of patch-sizes such that $|P| = o$. We set $r = 6\log pl + 14\ell$ (3,968 in our case), $r^* = 6\log pl^* + 14\ell$ (1,664 in our case) and $q = \frac{1}{o} \sum_{i=1}^o \frac{1}{P_i}$, the ratio of DReLUs left. The communication cost of ReLU and bReLU is defined as the communication that remains after DReLU and pDReLU have been applied. The communication cost of some typical values ($i = 128, o = 256, f = 3, h = 64, \ell = 64, \ell^* = 16, q = 0.1$) is shown. The complexity reduction caused by using identity channels is not shown. Finally, theoretically, in approximate DReLU layer, we can further reduce communication by decreasing the \mathbb{Z}_p field size, as the only requirement is that p is a prime such that: $p > 2 + \ell^*$.

The usage of bReLU layer does not decrease the number of communication rounds required in comparison to a standard ReLU layer. This holds true irrespective of whether or not approximate DReLU is utilized. As a result, the bReLU layer incurs a cost of 10 communication rounds. We refer the readers to (Wagh, Gupta, and Chandran 2019) for more details.

Secure Inference Performance



(a) Runtime

(b) Bandwidth

Components Contribution

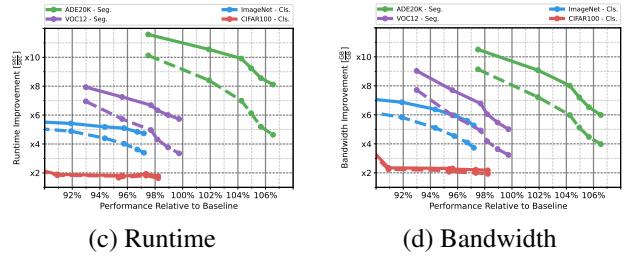


Figure 4: Secure Inference: Runtime (a) and Bandwidth consumption (b) improvement are depicted vs the task performance operating point. The baseline model comprises the pre-trained version with all ReLUs and full ReLU resolution. Our evaluation employs the SecureNN protocol, conducted on a cloud environment. Specific runtime and bandwidth consumption values are explicitly provided for operating points denoted with larger circles. Figures (c) and (d) show the contribution of each component to the Secure Inference runtime and bandwidth consumption metrics. Solid lines are evaluations with Approximate DReLU, while the dashed lines indicate the network with bReLUs evaluated without approximate DReLU.

	Layer	Communication	Typical Values
1	Conv2d _{h,i,f,o}	$h^2(2i+o)\ell + 2f^2oil$	21MB
2	DReLU _{h,o}	h^2ro	520MB
3	App. DReLU _{h,o}	h^2r^*o	218MB
4	pDReLU _{h,o,P}	h^2r^*qo	52MB
5	App. pDReLU _{h,o,P}	h^2r^*qo	22MB
6	ReLU _{h,o}	$5oh^2\ell$	42MB
7	bReLU _{h,o,P}	$(3+2q)oh^2\ell$	27MB
8	Conv2d+ReLU	(1) + (2) + (6)	583MB
9	Conv2d+bReLU	(1) + (5) + (7)	70MB

Table 2: Communication complexity: Communication complexity of Conv2D and ReLU layers. Our approximation (line (9)) requires almost an order of magnitude less communication bandwidth, compared to the baseline approach (line (8)). See discussion and details in the text.

Ablation Study

Alternative Patch Sizes Sets We investigate the effect on performance of using a different set of patch-sizes instead of the Knapsack optimal patch-sizes. Table 3 presents a comparison of three different patch size sets: (1) A set of naive 4×4 constant patch sizes, (2) Channel-shuffled Knapsack patch sizes, which are similar to Knapsack-optimized patch sizes but are shuffled among channels within the same layer, and (3) Knapsack-optimized patch sizes using the same DReLU budget as the constant version. The purpose of the third set is to differentiate between the respective contributions of the coarse-grained budget allocation across layers and the fine-grained budget allocation across channels within the same layer. We note that shuffling preserves the patch size layer distribution, and thus only partially isolate the contribution of this fine grained allocation.

Approximate DReLU Figures 4(c) + (d) breaks down the contribution of solely bReLUs (dashed lines) and adding the

	ImgNt ResN50	CIFR100 ResN18	AD20K MbNV2	VOC12 ResN50
4×4	60.59	59.07	31.06	67.64
KS-Shuffled	69.55	70.67	33.20	72.95
KS-Optimal	71.04	71.28	35.08	74.91

Table 3: Alternative Patch Size Sets The effect of using different sets of patch sizes at a given budget of DReLUs on task performance. *KS-shuffled* refers to shuffling the set of Knapsack optimal patch sizes within layers. *KS-Optimal* refers to using the set of Knapsack-optimal patch sizes.

Approximate DReLUs on top of them. Largely, both bReLUs and Approximate DReLU contribute to the metrics with bReLUs being more significant.

Conclusions

We propose a technique to decrease DReLU counts in Neural Networks, aiming to enhance Secure Inference runtime and bandwidth usage. Our key insight is that we can replace ReLU activations with Block-ReLU(bReLU) activations which operate on a patch. Utilizing our devised distortion measure, we formulate the problem of finding the optimal bReLU patch-size as a Knapsack optimization problem. Our formulation allows us to precompute distortions *once* and then reuse them to find the optimal bReLU patch sizes for different DReLU budgets. This flexibility allows us to explore the tradeoff between task performance and DReLU utilization.

We showcase our approach's competitiveness on the standard CIFAR-100 classification benchmark using ResNet18 and surpass the current state-of-the-art method, SNL, in ImageNet classification with ResNet50. We are the *first* to demonstrate Secure Inference on the Semantic Segmentation task. We evaluate runtime and bandwidth consumption for a secure inference setting implemented within the 3-party SecureNN protocol.

References

- Aflalo, Y.; Noy, A.; Lin, M.; Friedman, I.; and Zelnik, L. 2020. Knapsack pruning with inner distillation. *arXiv preprint arXiv:2002.08258*.
- Berger, T. 2003. Rate-distortion theory. *Wiley Encyclopedia of Telecommunications*.
- Chen, L.-C.; Papandreou, G.; Schroff, F.; and Adam, H. 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.
- Cho, M.; Joshi, A.; Reagen, B.; Garg, S.; and Hegde, C. 2022. Selective network linearization for efficient private inference. In *International Conference on Machine Learning*, 3947–3961. PMLR.
- Contributors, M. 2020a. MM Segmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. <https://github.com/open-mmlab/mmsegmentation>.
- Contributors, M. 2020b. OpenMMLab’s Image Classification Toolbox and Benchmark. <https://github.com/open-mmlab/mmclassification>.
- 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.
- Dubey, A.; Chatterjee, M.; and Ahuja, N. 2018. Coreset-based neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 454–470.
- Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88: 303–338.
- Ghodsi, Z.; Jha, N. K.; Reagen, B.; and Garg, S. 2021. Circa: Stochastic relus for private deep learning. *Advances in Neural Information Processing Systems*, 34: 2241–2252.
- Ghodsi, Z.; Veldanda, A. K.; Reagen, B.; and Garg, S. 2020. Cryptonas: Private inference on a relu budget. *Advances in Neural Information Processing Systems*, 33: 16961–16971.
- Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; and Wernsing, J. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, 201–210. PMLR.
- Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.
- Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.
- Hassibi, B.; and Stork, D. 1992. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5.
- 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.
- He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, 1389–1397.
- Helbitz, I.; and Avidan, S. 2021. Reducing ReLU count for privacy-preserving CNN speedup. *arXiv preprint arXiv:2101.11835*.
- Hesamifard, E.; Takabi, H.; Ghasemi, M.; and Wright, R. N. 2018. Privacy-preserving machine learning as a service. *Proc. Priv. Enhancing Technol.*, 2018(3): 123–142.
- Hubara, I.; Nahshan, Y.; Hanani, Y.; Banner, R.; and Soudry, D. 2021. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, 4466–4475. PMLR.
- Jha, N. K.; Ghodsi, Z.; Garg, S.; and Reagen, B. 2021. Deepreduce: Relu reduction for fast private inference. In *International Conference on Machine Learning*, 4839–4849. PMLR.
- Juvekar, C.; Vaikuntanathan, V.; and Chandrakasan, A. 2018. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, 1651–1669.
- Kamma, K.; and Wada, T. 2019. Reconstruction error aware pruning for accelerating neural networks. In *Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part I* 14, 59–72. Springer.
- Kellerer, H.; Pferschy, U.; Pisinger, D.; Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. The multiple-choice knapsack problem. *Knapsack Problems*, 317–347.
- Khan, T.; Bakas, A.; and Michalas, A. 2021. Blind faith: Privacy-preserving machine learning using function approximation. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, 1–7. IEEE.
- Krizhevsky, A.; Nair, V.; and Hinton, G. 2010. Cifar-10 (canadian institute for advanced research). *URL http://www.cs.toronto.edu/kriz/cifar.html*, 5(4): 1.
- Kumar, N.; Rathee, M.; Chandran, N.; Gupta, D.; Rastogi, A.; and Sharma, R. 2020. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*, 336–353. IEEE.
- Lam, S. K.; Pitrou, A.; and Seibert, S. 2015. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 1–6.
- Li, S.; Xue, K.; Zhu, B.; Ding, C.; Gao, X.; Wei, D.; and Wan, T. 2020. Falcon: A fourier transform based approach for fast and secure convolutional neural network predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8705–8714.
- 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.
- Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In

Proceedings of the IEEE international conference on computer vision, 5058–5066.

Mohassel, P.; and Rindal, P. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 35–52.

Mohassel, P.; and Zhang, Y. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, 19–38. IEEE.

Nishide, T.; and Ohta, K. 2007. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *Public Key Cryptography—PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography Beijing, China, April 16–20, 2007. Proceedings 10*, 343–360. Springer.

Patra, A.; Schneider, T.; Suresh, A.; and Yalame, H. 2021. {ABY2. 0}: Improved {Mixed-Protocol} Secure {Two-Party} Computation. In *30th USENIX Security Symposium (USENIX Security 21)*, 2165–2182.

Riazi, M. S.; Weinert, C.; Tkachenko, O.; Songhori, E. M.; Schneider, T.; and Koushanfar, F. 2018. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia conference on computer and communications security*, 707–721.

RoyChowdhury, A.; Sharma, P.; Learned-Miller, E.; and Roy, A. 2017. Reducing duplicate filters in deep neural networks. In *NIPS workshop on deep learning: Bridging theory and practice*, volume 1, 1.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Shen, M.; Yin, H.; Molchanov, P.; Mao, L.; Liu, J.; and Alvarez, J. M. 2022. Structural pruning via latency-saliency knapsack. *Advances in Neural Information Processing Systems*, 35: 12894–12908.

Son, S.; Nah, S.; and Lee, K. M. 2018. Clustering convolutional kernels to compress deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, 216–232.

Srinivasan, W. Z.; Akshayaram, P.; and Ada, P. R. 2019. DELPHI: A cryptographic inference service for neural networks. In *Proc. 29th USENIX Secur. Symp*, 2505–2522.

Wagh, S.; Gupta, D.; and Chandran, N. 2019. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proc. Priv. Enhancing Technol.*, 2019(3): 26–49.

Wagh, S.; Tople, S.; Benhamouda, F.; Kushilevitz, E.; Mittal, P.; and Rabin, T. 2020. Falcon: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint arXiv:2004.02229*.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 633–641.