Secure and Efficient Video Inferences with Compressed 3-Dimensional Deep Neural Networks

Bingyu Liu
Wentworth Institute of Technology
Boston, United States

Ali Arastehfard
University of Connecticut
Storrs, United States

Rujia Wang
Microsoft Research
Redmond, United States

Weiran Liu
Alibaba Group
Beijing, China

Zhongjie Ba
Zhejiang University
Hangzhou, China

Shanglin Zhou
Yuan Hong
University of Connecticut
Storrs, United States

Abstract

Deep neural network (DNN) services have been widely deployed for efficient and accurate inferences in many different domains. In practice, a client may send its private data (e.g., images, text messages and videos) to the service to get the inferences with the proprietary DNN models. Significant privacy and security concerns would emerge in such scenarios. Cryptographic inference systems have been proposed to address such privacy and security concerns. However, existing systems are tailored for DNNs on image inferences, but not directly applicable to video inference tasks that operate on the spatio-temporal (3D) features. To address such critical deficiencies, we design and implement the first cryptographic inference system, `Crypto3D`, which privately and efficiently infers videos with compressed 3D DNNs while ensuring rigorous privacy guarantees. We also update most cryptographic inference systems (designed for images) to support video understanding on 3D features with non-trivial extensions, treating them as baselines. We evaluate `Crypt3D` and benchmark with baselines utilizing the widely adopted C3D and I3D models on the UCF-101 and HMDB-51 datasets. Our results demonstrate that `Crypto3D` significantly outperforms existing systems on execution time: 554.68× vs. CryptoDL (3D), 189.21× vs. HEANN (3D) , 182.61× vs. MP-SPDZ (3D), 133.56× vs. E2DM (3D), 11.09× vs. Intel SGX (3D), 8.90× vs. Gazelle (3D), 3.71 × vs. Delphi (3D), 12.97 × vs. CryptFlow2 (3D), 1.49× vs. Cheetah (3D); accuracy: 82.4% vs. < 80% for all of them.[1]

CCS Concepts

• **Security and privacy → Privacy-preserving protocols**;

Keywords

privacy, neural network predictions, video inference

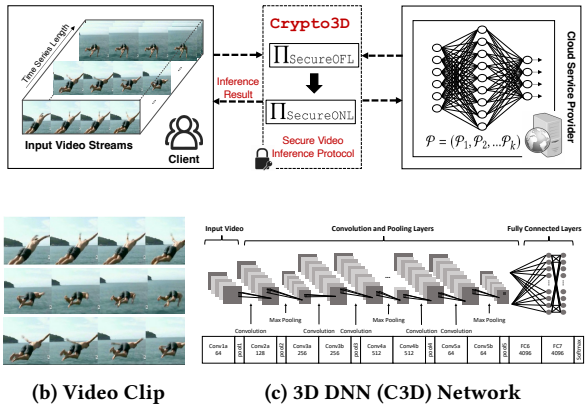
ACM Reference Format:

Bingyu Liu, Ali Arastehfard, Rujia Wang, Weiran Liu, Zhongjie Ba, Shanglin Zhou, and Yuan Hong. 2025. Secure and Efficient Video Inferences with Compressed 3-Dimensional Deep Neural Networks. In *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy (CODASPY '25), June 4–6, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3714393.3726505

[1]Code is available at https://github.com/datasec-lab/crypto3D

CODASPY '25, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-1476-4/2025/06
https://doi.org/10.1145/3714393.3726505



(b) Video Clip **(c) 3D DNN (C3D) Network**

Figure 1: `Crypto3D` for private inferences based on spatial-temporal (3D) features. Figure 1a shows the *Cryptographic inference* for video classification between a client C and a server $\mathcal{S}$. The client C holds the input video (Figure 1b), while the server $\mathcal{S}$ holds the pre-trained 3D DNN model (Figure 1c). Private inference is achieved through the two-party interactions via `Crypto3D`, ensuring that neither party can learn any private information from each other.

1 Introduction

Deep neural networks (DNNs) have seen a rising deployment in practice encompassing object detection, image and action classification, anomaly detection, among others. Within the framework of a client-server model for DNNs, such as deep learning as a service, the client typically transmits its data to a server. The server then furnishes inference services, including classification and prediction, utilizing its pre-trained DNN models. However, the data supplied by clients often contain substantial private information, such as human faces, activities, and workspace details. Directly disclosing them to the cloud would compromise users' privacy. On the contrary, the pre-trained DNN model should also be considered as proprietary information for the server and cannot be directly shared to clients for local inference.

To mitigate such privacy risks, cryptographic inference systems [4, 29, 56] are designed for *secure inferences* (see Table 1). A cryptographic inference protocol allows the client to input its private data in the encrypted form, and privately obtain the learning result from the provider. In this process, the server cannot learn anything about the inputs, while the client cannot obtain any information about the model weights, thus reducing privacy and security risks.

Several cryptographic primitives can be used to construct secure inference systems. Fully Homomorphic Encryption (FHE) [4] can provide strong privacy guarantees, but it is computationally expensive. Moreover, some non-polynomial functionalities (e.g., non-linear activation functions ReLU) cannot be directly supported by FHE. Garbled Circuit (GC) [56] and Secret Sharing (SS) [2, 9, 55] support arbitrary functionality, but GC results in significant communication overheads, while SS requires high round complexity. Thus, directly using such primitives is not ideal for secure DNN inferences.

Many existing works use one or more cryptographic primitives to construct secure inference systems, with specified optimizations for DNN model inferences. As shown in Table 1, most recent cryptographic inference systems are proposed to improve the performance (e.g., efficiency and accuracy) on inferring images with 2D features. Unfortunately, *securely inferring images based on 2D features* by state-of-the-art (SOTA) systems are far from enough for video-based applications. Such task poses new significant challenges from cryptographic systems, more complex model architecture, and their integration. Compared with the 2D ConvNets, most 3D ConvNets have to infuse the temporal information of the videos after each convolution/pooling operations. Performing 3D convolution and pooling operations is expected to deliver temporal information across all the neural network layers to the end. Integrated with both spatial and temporal information in each feature, 3D ConvNets (e.g., the recent C3D [48] and I3D [5] networks) have been demonstrated to be more accurate on video inferences than 2D ConvNets [5, 48]. However, to our best knowledge, cryptographic inferences based on 3D features for video DNNs have not been studied yet.

To address this critical deficiency, we design and implement the first cryptographic inference system `Crypto3D` for private inferences based on 3D spatial-temporal features (both C3D [48] and I3D [5]). `Crypto3D` enables to privately perform inferences for video classification, action recognition and prediction, as well as visual anomaly detection. We boost the efficiency of `Crypto3D` for a hybrid design (with cryptographic primitives of FHE, GC and SS) by adapting three new methods: (1) *optimizing the matrix operations for 3D DNN*, (2) *ciphertext packing technique*, and (3) *surrogate Lagrangian relaxation (SLR)-based network pruning [13] for the 3D DNNs*. Specifically, we make the following major contributions:

- To our best knowledge, we design and implement the first cryptographic inference system `Crypto3D` for private and accurate video inferences based on 3D DNNs.
- Given the high complexity of the 3D video models and the naturally incurred computational overheads, to boost the efficiency of `Crypto3D`, we also take the first step towards the co-design for harmonizing the cryptographic primitives (e.g., ciphertext packing), matrix operations optimization, and model compression (e.g., weight-pruning optimization) for 3D video models. We prove that the co-design in `Crypto3D` (for boosting efficiency) does not leak private information.
- We conduct substantial experiments while benchmarking with all non-trivially extended 3D cryptographic inference systems. To do so, we redesign and re-implement a wide variety of cryptographic systems for image inferences to support the video inferences. Such non-trivial extensions involve the complex tasks of tailoring their dimensions and formats to match different cryptographic primitives. We demonstrate that `Crypto3D` achieves superior performance over all the baselines.

2 Related Work

Homomorphic Encryption-based Protocols. Homomorphic encryption enables mathematical operations on ciphertext without requiring knowledge of the unencrypted data. One notable protocol, CryptoNets [12], proposes an HE-based secure neural network inference framework. Other applications of DNNs leverage faster homomorphic encryption schemes [4, 60], but these schemes have limitations in terms of the supported depth of encryption and the ability to perform multiplication operations without bootstrapping [60]. In CryptoDL, polynomials are employed to approximate complex nonlinear activation functions, such as Sigmoid and tanh. However, it is important to note that homomorphic encryption is not ideal and practical in terms of efficiency due to significant computational overhead.

MPC-based Protocols. MPC enables multiple parties to jointly evaluate a function without revealing their individual inputs to each other, except for the final results. Existing works include Garbled Circuit (GC) [1, 41, 43], Secret Sharing [2, 9, 55], and Mixed Protocol approaches [23, 34, 42]. For instance, [41] and [1] propose optimizations for neural network activation functions using garbled circuits. In [1], practical data aggregation protocols are designed based on Shamir's t-out-of-n secret sharing protocol [15]. However, secret sharing and garbled circuits have limitations that can introduce computational overhead. To address this, Mixed Protocols [23, 34, 42] have been proposed, which combine additive secret sharing or homomorphic encryption for linear operations and garbled circuits for non-linear computations. Delphi [33] builds upon Gazelle and improves it by incorporating garbled circuits and quadratic polynomials for activation functions. CrypTFlow2 [40] designs new two-party computation (2PC) protocols for secure comparison and division, aiming to balance round and communication complexity for secure inference tasks. Cheetah [19] presents a highly optimized architecture based on HE and communication-efficient primitives to handle the large overhead of the current 2PC-NN. However, these prior works are primarily focused on 2D ConvNets, and the inference results do not retain temporal information for video data. In `Crypto3D`, we employ the C3D model for performing cryptographic inference in video classification, thereby preserving the temporal features in the prediction.

TEE-based Protocols. Trusted Execution Environments (TEE) [6, 16, 25, 46, 47] provide a secure enclave where the model/data owner can isolate sensitive computations for DNN models from an untrusted software stack. TEEs ensure both data privacy and

Table 1: Comparison of cryptographic inference systems. Visor [38] provides confidentiality for video analysis via TEE. However, it still privately infers data (e.g., object detection and tracking) based on 2D features. PPVC [36] preserves privacy for video classification based on cryptographic protocols, but it still utilizes the 2D ConvNets without fully preserving temporal info.

Cryptographic Inference Systems	Design	Security	Video	Spatial	Temporal
CryptoNets [12]	HE	Cryptographic	✗	✓	✗
CryptoDL [17]	HE	Cryptographic	✗	✓	✗
XONN [41], MiniONN [29]	GC	Cryptographic	✗	✓	✗
DeepSecure [43]	GC	Cryptographic	✗	✓	✗
PSA [2], SPDZ [29]	SS	Cryptographic	✗	✓	✗
MLCapsule [16], Privado [46], Slalom [47]	TEE	Hardware-based	✗	✓	✗
Visor [38]	TEE	Hardware-based	✓	✓	✗
ABY3 [34], Crypten [26], CrypTFlow2 [40]	Mixed	Cryptographic	✗	✓	✗
GALA [58] Chameleon [42]	Mixed	Cryptographic	✗	✓	✗
Delphi [33], Cheetah [19]	Mixed	Cryptographic	✗	✓	✗
PPVC [36]	Mixed	Cryptographic	✓	✓	✗
Crypto3D (Ours)	Mixed	Cryptographic	✓	✓	✓

integrity. In[38], Visor is a proposed system that enables privacy-preserving video analytics services using a hybrid TEE architecture. It ensures strong confidentiality and integrity for video streams. TEE-based secure cryptographic inference often outperforms MPC protocols. However, it requires trust in the hardware, has a weaker threat model, and necessitates implementation within the enclave. Additionally, the vulnerability to side-channel attacks is a significant concern that needs to be addressed.

Differential Privacy-based Solutions. Differential privacy-based techniques for DNNs aim to reduce the amount of sensitive information carried by the data and mitigate the errors of noise addition on training. Shokri et al. [44] utilize differential privacy in deep learning models to ensure that data privacy is not compromised when sharing local parameters with the server. Other works [37, 50] propose different approaches to handle the trade-off between privacy and accuracy (i.e., adding noise to the weights [37] or dynamically setting the privacy budget [57]).

3 Preliminaries

3D CNN Neural Network. Given a video $\mathcal{V}$, we possess the following steps for the inference. First, the video is divided into multiple segments. Then, several frames from each part are selected to compose a clip. Then, these clips, representing the entire video, are fed into the 3D-CNNs respectively. The 3D-CNN, extending the 2D-CNN into the temporal dimension, is more adept at capturing the three-dimensional data features of videos. For C3D, it consists of $3 \times 3 \times 3$ convolutional kernels followed by $2 \times 2 \times 2$ pooling at each layer (as shown in Fig 2). The C3D model is trained on a large scale video dataset such as UCF101 [45] and Sports 1M [24]. For the generic feature extraction, the 3D convolutions are able to extract both spatial and temporal components information in the videos, e.g., the motion of objects, human action and human-object interactions. It directly encodes the temporal structure with a 3D convolutional network instead of 2D. The involved 3D kernel is able to extract information from both spatial and temporal dimensions and fuse them into the same feature [48]. Compared with a 2D ConvNet, a 3D ConvNet provides better modeled temporal information with 3D convolution and 3D pooling operations for more accurate video recognition. I3D [5] is a new Two-Stream Inflated 3D ConvNet based on the 2D ConvNet inflation. It enables

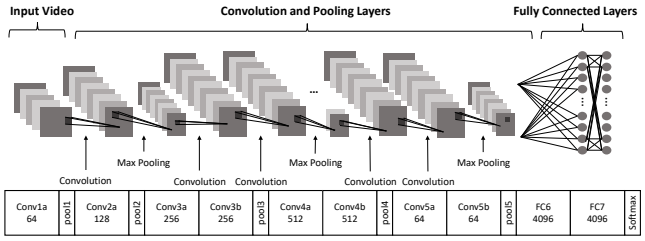


Figure 2: Illustration of the spatio-temporal convolution Network Architecture of C3D. 3D ConvNet is designed to have 8 convolution layers, 5 pooling layers, followed by 2 fully connected layers, and a softmax output layer.

Secure 3D Inference. We assume the generic two-party secure inference setting, involving a client C and a server $\mathcal{S}$. The pre-trained 3D neural network model is held by the server $\mathcal{S}$ while the input video to be classified is held by the client C. The DNN architecture (i.e., dimensions and type of each layer in the neural networks) are known to the public. We consider the privacy of input video and the security of model weights during the inference process. We assume that the pre-trained DNN model from the server will not be changed and updated. This can be naturally extended to the updated variant with necessary parameters synchronization.

3.1 Cryptographic Primitives

Homomorphic Encryption. A homomorphic encryption of x enables the computing encryption of $f(x)$ without any knowledge of the decryption key. A *Linearly homomorphic public-key encryption* [11, 35] with a finite ring $\mathcal{R}$ as the message space includes a set of probabilistic polynomial-time algorithms $\pi_{\text{HE}} = (\text{KGen}, \text{Enc}, \text{Dec}, \text{Eval})$:

- $(\text{pk}, \text{sk}) \leftarrow \text{HE.KGen}$. The key generation algorithm is used to generate a public/private key pair (pk, sk).

- $c \leftarrow \text{HE.Enc}(\text{pk}, m)$. The ciphertext c is generated by the encryption algorithm with the public key pk and the message m.
- $m \leftarrow \text{HE.Dec}(\text{sk}, c)$. The message m can be obtained by running the decryption algorithm with the secret key sk and the ciphertext c.
- $c_l \leftarrow \text{HE.Eval}(\text{pk}, c_1, c_2, \mathcal{L})$. The new ciphertext c_l is generated by the evaluation algorithm with pk, two encrypted messages c_1, c_2, and the linear function $\mathcal{L}$, where $\mathcal{L}$ maps (m_1, m_2) to $km_1 + m_2$ for $k \in \mathcal{R}$.

This work involves several different HE libraries for implementations. More details are deferred to the Table 3.

Oblivious Transfer. Oblivious Transfer (OT) [39] is a fundamental cryptographic building block in MPC. OT is executed between a sender and a receiver. The sender has two inputs x_0, x_1 while the receiver wants to receive the x_b (a selection bit b) without revealing b or learning anything from the server. In this work, we use $(\perp; x_b) \leftarrow \text{OT}(x_0, x_1; b)$ to represent this functionality.

Garbled Circuits. Garbled Circuits (GC), proposed by Yao [56], is the first secure two-party computation protocols support computations on arbitrary functions. The garbled circuit generator (one party) prepares the encrypted circuit computing f while the garbled circuit evaluator (the other party) computes the output of the circuit without learning any intermediate values. Denoting the Boolean circuit as C, for the input $\mathbf{x}$, a *Garbling scheme* includes a group of algorithms $GS = (\text{Garble}, \text{Eval})$, as follows:

- $(\widetilde{C}, \{\text{lab}_{i,0}, \text{lab}_{i,1}\}_{i\in[n]}) \leftarrow \text{GS.Garble}(C)$. Given the input of a boolean circuit C, the Garble algorithm outputs a garbled circuit $\widetilde{C}$ and a set of labels $\{\text{lab}_{i,0}, \text{lab}_{i,1}\}_{i\in[n]}$, where $\text{lab}_{i,b}$ is the assigned value $b \in \{0, 1\}$ to the i-th input label.
- $y \leftarrow \text{GS.Eval}(\widetilde{C}, \{\text{lab}_{i,x_i}\})$. The evaluation algorithm outputs $y = C(x)$ with the input garbled circuit $\widetilde{C}$ and the given labels $\{\text{lab}_{i,x_i}\}$ corresponding to the input $x_i \in \{0, 1\}$ for $i \in [n]$.

4 Cryptographic Inference Protocol Co-Design

We define the DNNs model owned by server privately as $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{|N|})$ with k layers. $\mathcal{P}_i$ represents the layer of the model. Given a video $\mathcal{V} = \{f_i\}_{i=1}^{i=N}$ consisting of N frames. Video-level takes a sequence consisting of multiple frames as input, we have $\mathcal{V} = \{f_{t_1}, f_{t_2}, ..., f_{t_n}\}$, where $t_1 \le t_2 \le t_3, ..., \le t_{n<N}$.

In this section, we first define the threat model and the security guarantees for our secure inference system `Crypto3D`, and then illustrate the protocol design.

4.1 Threat Model

We consider the security of `Crypto3D` under the semi-honest model. A protocol is secure against semi-honest adversaries if the corrupted parties in the real world have views that are indistinguishable from their views in the ideal world. We refer to the ideal-world adversary as simulator π_{Sim}, since it generates a real-world view while in the ideal world. Showing that such simulator exists proves that there is nothing an adversary can accomplish in the real world that could not also be done in the ideal world. More specifically, we denote the protocol $\pi_{\text{SecureINF}}$, the polynomial-time functionality $\mathcal{F}$, views of party View_S, final output of party $\mathbf{y}$ and the corrupted parties $\widetilde{P}$. **Sim** denotes a simulator algorithm. Then we have $\pi_{\text{SecureINF}}(\widetilde{P_i}; \mathbf{x})^{\text{Real}}$, which represents each party runs the protocol honestly with given private input $\mathbf{x}$. In this case, the output is $\{\text{View}_S^{\pi_{\text{SecureINF}}} | i \in \widetilde{P_i}\}, (\mathbf{y})$. Similarly, we denote $\text{Sim}_{\mathcal{F}}(\widetilde{P_i}; \mathbf{x})^{\text{Ideal}}$ to compute the $\mathcal{F}(\mathbf{x})$, the output would be $\text{Sim}(\widetilde{P_i}, \{(\mathbf{x}, \mathbf{y}) | i \in \widetilde{P_i}\}, (\mathbf{y})$.

Definition 4.1. (Security w.r.t semi-honest behavior): A cryptographic inference protocol $\pi_{\text{SecureINF}}$ between the two parties C and $\mathcal{S}$ with input feature vector $\mathbf{x}$ and the pre-trained model parameters $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ...\mathcal{P}_{|N|})$ securely computes the probabilistic polynomial-time functionality $\mathcal{F}$, and satisfies the Correctness and Security.

- *Correctness*: For all set of model parameters $\mathcal{P}$ and all feature input vectors $\mathbf{x}$, the output at the end of protocol is the correct prediction $\mathbf{y}$ in the cryptographic inference.
- *Security*:
  - Semi-Honest Server Security. There exists a simulator Sim_S such that $\{\text{View}_S^{\pi_{\text{SecureINF}}}(\mathcal{P}, \mathbf{x})\}, (\mathbf{y})$ $\approx_c \{\text{Sim}_S(\mathcal{P}, \mathcal{F}(\mathcal{P}, \mathbf{x}))\}, (\mathbf{y})$, where $\text{View}_S^{\pi_{\text{SecureINF}}}$ denotes the view of the server in the protocol $\pi_{\text{SecureINF}}$. Sim_S is able to simulate a view of the semi-honest server without learning any private input vector $\mathbf{x}$ of client in polynomial time.
  - Semi-Honest Client Security: There exists a simulator Sim_C such that $\text{View}_C^{\pi_{\text{SecureINF}}}(\mathcal{P}, \mathbf{x}) \approx_c \text{Sim}_C(\mathbf{x}, \mathcal{F}(\mathcal{P}, \mathbf{x}))$, where $\text{View}_C^{\pi_{\text{SecureINF}}}$ denotes the view of the client in the protocol π and output represents the results of inference. Sim_C is able to simulate a view of the semi-honest client without learning any pre-trained model parameters $\mathcal{P}$ in polynomial time.

4.2 Protocol Design

Without loss of generality, we present the design of `Crypto3D` based on C3D [48]. This design can be extended, as demonstrated in the extended evaluations for another 3D DNN model I3D, in Section 6.

The design of `Crypto3D` is formally shown in Figure 3. It contains two sub-protocols $\pi_{\text{SecureLIN}}$ and $\pi_{\text{SecureNonLIN}}$, details of both are shown in Figure 4 and Figure 5.

Protocol ($\pi_{\text{SecureLIN}}$). `Crypto3D` provides secure computation for linear layers. First, (pk, sk) can be fetched via the KGen algorithm for the client. We denote $\langle r_i \rangle \leftarrow \mathbb{R}^{\boldsymbol{n}}, i \in [1, .., N]$ and $\langle s_i \rangle \leftarrow \mathbb{R}^{\boldsymbol{n}}, i \in [1, .., N]$ as the random masking vectors for the i-th layer. In the linear layer, the encrypted ciphertext $\text{CT}_i^{\mathcal{S}} \leftarrow \text{Enc}(\text{pk}, \langle r_i \rangle)$ is sent to the server by the client. With the Eval procedure, the server computes the O_i^{LIN} and send its ciphertext back to the client. Then, the client decrypts and learns $\langle \mathcal{P}_i r_i \rangle$. Thus, the additive secret sharing of $\mathcal{P}_i \cdot [\![r_i]\!]$ is held by both the client and the server before the online phase execution. Given the input $\mathbf{x}$ ($\mathbf{x} \in \mathbb{Z}^{C\times D\times H\times W} \leftarrow \text{Process}(\mathcal{V})$), the server receives $\mathbf{x} - \langle r_1 \rangle$. At this time, the additive secret shares of $\mathbf{x}$ are held by the client and server, respectively. At the beginning of the i-th layer evaluation, $\mathbf{x}_i$ can be fetched from the first $(i-1)$ layers of the neural network. The client holds $\langle r_i \rangle$ while server holds $\langle x_i \rangle - \langle r_i \rangle$. For the evaluation of the linear

Protocol $\pi_{\texttt{SecureINF}}(\mathcal{P}_i, \mathbf{x}, r, s)$:

Input: DNNs model $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{|N|})$ and Video $\mathcal{V}$
Output: Inference results $\mathcal{O}$

$\{f_1, f_2, f_3, ..., f_k\} \leftarrow \mathcal{V}$
$\mathcal{V} = \{f_{t_1}, f_{t_2}, ..., f_{t_n}\}$, where $t_1 \leq t_2 \leq t_3, ..., \leq t_{n<N}$
$\mathbf{x} \in \mathbb{Z}^{C\times D\times H\times W} \leftarrow \text{Process}(\mathcal{V})$
foreach $i \in [1, |N|]$ **do**
  switch $\mathcal{P}_i$ **do**
    case Linear :
      $\mathcal{O}_i^{\text{LIN}} \leftarrow \pi_{\texttt{SecureLin}}()$
    case Non-Linear :
      $\mathcal{O}_i^{\text{NonLIN}} \pi_{\texttt{SecureNonLIN}}()$
return $\mathcal{O} \leftarrow \mathbf{x}_N - \langle r_N \rangle$

Figure 3: Protocol $\pi_{\texttt{SecureINF}}$

Protocol $\pi_{\texttt{SecureNonLIN}}()$:

Input: DNNs model $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{|N|})$ and Video $\mathcal{V}$
Output: $\mathcal{O}_s^{\text{NonLIN}}$ and $\mathcal{O}_c^{\text{NonLIN}}$

$\mathcal{S}$ computes $\langle x_i - \langle r_i \rangle \rangle$
/ set the x_i to be the result of evaluation for the first (i-1)-th layers on x
$x_i \leftarrow \text{EVAL}_{i-1}(\mathbf{x})$
foreach $i \in [1, N]$ **do**
  switch $\mathcal{P}_i$ **do**
    case Garbled circuits :
      / Construct Garbled Circuit
      $\{\widetilde{C}, lab_{i,0}, lab_{i,1}\}_{i\in[n]} \leftarrow \text{GS.Garble}(1^k, C)$
      Invoke OT$(\langle \mathrm{r_{i+1}} \rangle, \langle \mathrm{P_i} \cdot \mathrm{r_i} - \mathrm{s_i} \rangle)$
      $\mathcal{O}_s^{\text{NonLIN}} \leftarrow$ Compute $\langle \mathbf{x}_{i+1} - \langle r_{i+1} \rangle \rangle_s$
    case Polynomial approximation :
      / Compute Beavers triples
      $\langle \mathbf{x}_{i+1} \rangle_1, \langle \mathbf{x}_{i+1} \rangle_2 \leftarrow \text{Beaverstriples}()$
      $\mathcal{O}_c^{\text{NonLIN}} \leftarrow$ Computes$\langle x_{i+1} \rangle_1 - \langle r_{i+1} \rangle$
      $\mathcal{O}_s^{\text{NonLIN}} \leftarrow$ Computes $\mathcal{O}_c^{\text{NonLIN}} + \langle x_{i+1} \rangle_2$

Figure 4: Protocol $\pi_{\texttt{SecureNonLIN}}$

layer(s), the server computes $\mathcal{P}_i \cdot (\mathbf{x}_i - \langle r_i \rangle)$ via the Permu$(\cdot)$ (via Equation 2), which ensures that the additive shared secrets of $\mathcal{P}_i \cdot \mathbf{x}_i$ are held by the client and server.

Protocol ($\pi_{\texttt{SecureNonLIN}}$). Regarding the non-linear layer execution, the execution of activation function depends on what type of function. The garbled circuit $\widetilde{C}$ is constructed via GC schemes. It helps to solve the ReLU function by exchanging the labels for input wires with $\langle r_{i+1} \rangle$ and $\mathcal{P}_i \cdot \langle r_i \rangle - \langle s_i \rangle$.

On the other hand, the Beaver's multiplication procedure is executed for the polynomial approximation evaluation. The client and sever will hold the $\langle x_{i+1} \rangle_1$ and $\langle x_{i+1} \rangle_2$, separately after the Beaver's multiplication procedure. At this time, the client sends the results of the $\langle x_{i+1} \rangle_1 - \langle r_{i+1} \rangle$ to the server. The $\langle x_{i+1} \rangle - \langle r_{i+1} \rangle$ will

Protocol $\pi_{\texttt{SecureLIN}}()$:

Input: DNNs model $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{|N|})$ and Video $\mathcal{V}$
Output: Encode$(\langle \text{Permu}(\mathcal{P}_i) \rangle \cdot \langle r_i \rangle), \langle r_i \rangle$
Encode$(\langle \text{Permu}(\mathcal{P}_i) \rangle \cdot \langle r_i \rangle_S$

$(\text{pk}, \text{sk}) \leftarrow \text{KGen}(1^\lambda)$
$\langle r_i \rangle, \langle s_i \rangle \leftarrow \mathbb{R}^n (\forall i \in [1, .., |N|])$
foreach $i \in [|N| - 1]$,
  $\text{CT}_i^S \leftarrow \text{Enc}(\text{pk}, \langle r_i \rangle)$
  $\mathcal{O}_i^{\text{LIN}} \leftarrow \text{Eval}_S((\text{pk}, \text{Encode}\langle \text{Permu}(\mathcal{P}_i) \rangle \cdot \langle r_i \rangle - \langle s_i \rangle$
  $\mathcal{S}$: learns $\langle s_i \rangle, \langle \mathcal{P}_i r_i \rangle$
  $\mathcal{C}$: learns $\langle \mathcal{P}_i r_i \rangle$

Figure 5: Protocol $\pi_{\texttt{SecureLIN}}$

be obtained by adding the $\langle x_{i+1} \rangle_2$. Finally, the client learns the x_N from the received x_N - $\langle r_N \rangle$.

Matrix-Vector Multiplication. We assume that the input matrix $\mathcal{P}$ has the size $n_0 \times n_i$, where n_i is smaller than the number of plaintext slots n_s. We denote the sub-matrices $\mathcal{P}_{ij}$ (where $0 \leq i < n_0$ and $0 \leq j < l$) with the size of $1 \times (n_i/l)$, which is splited from the $\mathcal{P}$. Next we pack the different matrics $(l \cdot n_s)/n_i$ into a single ciphertext, and the $n_c = (n_s/n_i)$ copies of the input vector r into a single cipertext. With the encoding n_c, the first diagonal vectors of the matrix into a plaintext vector as below:

$$((\mathcal{P}_{0,0}|\mathcal{P}_{1,1}|...|\mathcal{P}_{l-1,l-1})|(\mathcal{P}_{l,0}|\mathcal{P}_{l+1,1}|...|\mathcal{P}_{2l-1,l-1})|...$$
$$(\mathcal{P}_{l\cdot(n_c-1),0}|\mathcal{P}_{l\cdot(n_c-1)+1,1}\cdots|\mathcal{P}_{l\cdot(n_c-1)+l-1,l-1})) \in \mathbb{R}^{n_s}$$

Each extended diagonal vector is encrypted in a single ciphertext and these ciphertexts are multiplied with l rotations of the encrypted vector r. Next we add together and the output (ciphertext) represents (n_i/l)- sized $(l \cdot n_c)$ chunks. With the $\log(n_i/l)$ rotations, we get the ciphertext with the first $(l \cdot n_c)$ entries of $\mathcal{P} \cdot r \in \mathcal{R}^{n_0}$. Finally, we get the $n_0/(l \cdot n_c)$ ciphertexts after repeating the procedures $n_0/(l \cdot n_c)$.

Optimized Matrix Multiplication. Arithmetic operations of the encrypted matrices can lead to inefficiency in high-dimensional data tensors computation. To mitigate this issue, our Crypto3D utilizes the optimized matrix permutation [22] to efficiently perform the operation of matrix computations with ciphertext packing and parallelism [7]. The operation of the matrix multiplication can be considered as the sum of component-wise products with the specific permutations of the matrices themselves. We assume that there are two square matrices with size $n \times n$, the n permutations of the matrix A via the followings symmetric permutations:

$$\begin{aligned} \sigma(A)_{i,j} &= A_{i,i+j}, \tau(A) = A_{i+j,j} \\ \phi(A)_{j,j} &= A_{i,j+1}, \psi(A) = A_{i+1,j} \end{aligned} \tag{1}$$

where ϕ and ψ are denoted as the shifting functions for column and row, respectively. Then, the multiplication of two matrices (we denote A and B) with the order d can be computed as below:

$$A \cdot B = \sum_{k=1}^{d-1} (\phi^k \odot \sigma(A)) \times (\psi^k \odot \tau(B)) \tag{2}$$

where $\odot$ refers to the component-wise product and k represents the number of times for perturbation. As such, we can efficiently compute the two matrix multiplications. In Crypto3D, we utilize the function $\mathsf{Permu}(\cdot)$ to represent the computation of the n permutation operations. To boost the efficiency, we also utilize the vectorable HE "Ciphetext packing". We use the $\mathsf{Encode}(\cdot)$ to refer to the matrix transformations, which transforms a matrix into a plaintext vector with encoding map functions. Similarly, $\mathsf{Decode}(\cdot)$ is used for the plaintext vector transformations to the matrix. Equation 2 can be securely computed with the multiplicative property of HE. Our Crypto3D uses the optimized matrix multiplication and ciphertext packing [22] for the efficiency improvement. Since we can pack all the inputs into a single ciphertext and perform layer computation (e.g., convolutions) in parallel, we can enable the SIMD parallelism with the ciphertext packing.

Network Pruning. SOTA DNNs often suffer from challenges due to their large model sizes, which encompass millions of parameters. This characteristic results in extended inference times, substantial memory requirements, and poses significant difficulties in meeting critical requirements during the inference phase, such as real-time processing and low power consumption. The objective of deep model compression is to optimize the model in a more efficient format by alleviating the cost of the large model size and leave minimal impact on the performance of the model. Recently, there have been many orthogonal network optimization methods, such as ReLU optimizations [8, 20, 21, 28]. In our work, we consider the weight pruning (Irregular Pruning) [13] as the main optimization technique. In the future, we may continue our research work on ReLU optimizations.

In Crypto3D, we use DNN weight pruning [13], which aims to reduce the number of non-zero elements in the weight matrix [61]. Specifically, we consider the model compression technique *Surrogate Lagrangian Relaxation (SLR)* for weight pruning. For an N-layer DNN, where $i \in \{1, 2, ..., N\}$, we denote the weights at each convolutional layer as $\mathbf{W}_i$. The objective of irregular weight pruning is to minimize the DNN loss function while satisfy the constraints that the number of nonzero weights in each $\mathbf{W}_i$ should be less than the predefined percentage l_i: $\min_{\mathbf{W}_i} f(\mathbf{W}_i)\ s.t.\ \mathrm{card}(\mathbf{W}_i) < l_i$. The unconstrained forms can be written as below:

$$\min_{\mathbf{W}_i} f(\mathbf{W}_i) + \sum_{i=1}^{N} g_i(\mathbf{W}_i) \text{ where } g_i(\mathbf{W}_i) = \begin{cases} 0 & \text{if } \mathrm{card}(\mathbf{W}_i) \le l_i \\ +\infty & \text{otherwise} \end{cases} \tag{3}$$

In the equation, $f(\cdot)$ represents the nonlinear loss function, $g_i(\cdot)$ represents the non-differentiable "cardinality" penalty term for each layer, which is the indicator function [59]. In the equation, $f(\cdot)$ represents the nonlinear loss function, $g_i(\cdot)$ represents the non-differentiable "cardinality" penalty term for each layer, which is the indicator function [59]. In the SLR-based weight pruning, duplicate variables $\mathbf{Z}_i$ are introduced to decompose the loss function [3], and the problem is equivalently as $\min_{\mathbf{W}_i} f(\mathbf{W}_i) + \sum_{i=1}^{N} g_i(\mathbf{Z}_i), s.t. \mathbf{W}_i = \mathbf{Z}_i$. Lagrangian multipliers $\mathbf{\Lambda}_i$ are leveraged to relax the constraints, and quadratic penalties are used to penalize their violations. The result Augmented Lagrangian function can be written as Eq. 4, where $\|\cdot\|_F$ denotes the Frobenius norm and $\mathrm{tr}(\cdot)$ denotes the trace. This relaxed problem can be decomposed into two sub-problems and solved iteratively until convergence.

$$\begin{aligned} L_\rho(\mathbf{W}_i, \mathbf{Z}_i, \mathbf{\Lambda}_i) = f(\mathbf{W}_i) + \sum_{i=1}^{N} g_i(\mathbf{Z}_i) + \sum_{i=1}^{N} \mathrm{tr}\left[\mathbf{\Lambda}_i^T (\mathbf{W}_i - \mathbf{Z}_i)\right] \\ + \sum_{i=1}^{N} \frac{\rho}{2} \|\mathbf{W}_i - \mathbf{Z}_i\|_F^2 \end{aligned} \tag{4}$$

At iteration t, the first sub-problem is using SGD to minimize $L_\rho\left(\mathbf{W}_i, \mathbf{Z}_i^{t-1}, \mathbf{\Lambda}_i^t\right)$ for $\mathbf{W}_i$, while keeping $\mathbf{Z}_i = \mathbf{Z}_i^{t-1}$ for given values of multipliers $\mathbf{\Lambda}_i^t$ under the surrogate optimality condition $L_\rho\left(\mathbf{W}_i^t, \mathbf{Z}_i^{t-1}, \mathbf{\Lambda}_i^t\right) < L_\rho\left(\mathbf{W}_i^{t-1}, \mathbf{Z}_i^{t-1}, \mathbf{\Lambda}_i^t\right)$. The second sub-problem is minimizing $L_\rho\left(\mathbf{W}_i^t, \mathbf{Z}_i, \mathbf{\Lambda}_i^{t+1}\right)$ for $\mathbf{Z}_i$ by using projections onto discrete subspace. This step fixes $\mathbf{W}_i$ and analytically obtain the $\mathbf{Z}_i$. The surrogate optimality condition that need to be satisfied in this step is $L_\rho\left(\mathbf{W}_i^t, \mathbf{Z}_i^t, \mathbf{\Lambda}_i^{t+1}\right) < L_\rho\left(\mathbf{W}_i^t, \mathbf{Z}_i^{t-1}, \mathbf{\Lambda}_i^{t+1}\right)$.

We note that SLR also brings helps for engineering. The 3D video sequence is significantly more data-intensive than 2D images since the dimension of data is increased. This means that more data needs to be stored and processed during the training phase. In our implementation, we use a GPU-accelerated library for convolution evaluations to speedup the performance. Using GPU requires copying the layer weights and input into GPU RAM and then copying the output back into the CPU RAM. Thanks to SLR, the model size is reduced so that the 3D model can be fully loaded and copied during the entire training phase.

5 Security Analysis

We conduct the security analysis for the two cases where one of the parties is corrupted.

Theorem 5.1. *The secure two-party inference protocol $\pi_{\mathsf{SecureINF}}$ for Crypto3D (including $\pi_{\mathsf{SecureLIN}}$ and $\pi_{\mathsf{SecureNonLIN}}$ as shown in Figure 5 and 4) is secure against semi-honest adversaries.*

Proof. Our security proof follows the ideal-world/real-world paradigm. Our goal is to show that the adversary's view in real-world is indistinguishable to that in the ideal-world. Therefore, we prove this theorem by considering two cases separately: (1) Security against a semi-honest client, and (2) Security against a semi-honest server. Then, we build polynomial simulators Sim to simulate the views of all the participants of the protocol, detailed as below.

Case I: Client C is corrupted ($C \in \widetilde{P}_i$). In this case, we provide the security against the semi-honest client by constructing an ideal-world simulator Sim. We begin by describing the simulator $\widetilde{\mathcal{S}}$ as:

(1) Upon input, $\widetilde{\mathcal{S}}$ uniformly choose the random-tape r_i for the client C. During the offline $\pi_{\mathsf{SecureOFL}}$ phase, we have simulator $\widetilde{\mathcal{S}}$ receives pk and ciphertext $C(r_i) \leftarrow \mathsf{Enc}(\mathsf{pk}, r_i)$.

(2) Then, simulator $\widetilde{\mathcal{S}}$ sends ciphertext $C(s_i)' \leftarrow \mathsf{Enc}(\mathsf{pk}, -s_i')$ with random $s_i' \in \mathbb{R}^n$ to the server. Simulator $\widetilde{\mathcal{S}}$ invokes the $\widetilde{\mathcal{S}}_{GS}$ for garble circuits and runs on 1^λ and $1^{|C|}$ and sets the random value for the circuit $\widetilde{\mathcal{S}}_{GS}$ output $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$. For the i-th OT execution, $\widetilde{\mathcal{S}}$ sends the input $(\{L_{i,x_i}\}_{i\in[n]}\}$,

HYBRID Experiments $H_i^c(.)$

HYBRID^0 : It corresponds to real world distribution with the actual input matrices $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ...\mathcal{P}_{|N|})$ from the server.

HYBRID^1 : The simulator $\widetilde{\mathcal{S}}$ sends $\mathbf{y} - \langle r_N \rangle$. With the knowledge of the client's random tape, the simulator $\widetilde{\mathcal{S}}$ begins the evaluation of the i-th layer with $\mathbf{x} - \langle r_i \rangle$.The distribution on the view of the $\mathcal{A}$ for above is identical with this syntactic change.

HYBRID^2 : The server provides $\{L_{i,x_i}\}_{i\in[n]}\}$to replace the labels corresponding to 0 and 1 in each OT execution, where x_i is inputted from from the client in that OT execution. b is a result of setting the random tape and learning the input of corrupted client. Hybrid2 is indistinguishable from Hybrid1 with the sender security of OT execution.

HYBRID^3 : We generate $\widetilde{C}$ using the $\widetilde{\mathcal{S}}_{GS}$ on input $1^\lambda, 1^{|C|}$ and $C(z)$, where z is the input corresponding to the circuits evaluation. $C(z)$is an one-time pad (OTP) encryption, which is distributed identically to a random string. HYBRID3 is indistinguishable from Hybrid2 with the followed security of the garbled circuits.

HYBRID^4 : The multiplication triples in the offline phase is generated with the corresponding simulator $\widetilde{\mathcal{S}}$ for Beaver's protocol This follows from the simulation security that Hybrid4 is indistinguishable from Hybrid3.

HYBRID^5 : We use simulator $\widetilde{\mathcal{S}}$ for the Beaver's multiplication procedure for every quadratic approximation layer. Note that in this hybrid, $x_i - \langle r_i \rangle, \langle s_i \rangle$ and matrix $\mathcal{P}_i$ are no longer used for i-th layer evaluation. Similarly, this follows from the simulation security that Hybrid5 is indistinguishable from Hybrid4.

HYBRID^6 : The simulator $\widetilde{\mathcal{S}}$ is used for the function privacy with respect to each homomorphic evaluation in the offline phase. Also, $\widetilde{\mathcal{S}}$ only requires the $\mathcal{P}_i \cdot \langle r_i \rangle - \langle s_i \rangle$for the homomorphically evaluated ciphertext generation. This follows the function privacy of HE in which Hybrid6 is computationally indistinguishable from Hybrid5.

HYBRID^7 : We set input $-\left\langle s_i^{'} \right\rangle$ instead of the true value $(\mathcal{P}_i \cdot \langle r_i \rangle - \langle s_i \rangle)$. It is given to the $\widetilde{\mathcal{S}}$ with randomly sampled $\left\langle s_i^{'} \right\rangle$ from $\mathcal{R}^n$. The s_i is chosen uniformly at random, this indicates that the Hybrid7 is identically distributed to Hybrid6. Eventually, we note that Hybrid7 is identically distributed to the simulator $\widetilde{\mathcal{S}}'$soutput.

Figure 6: Random experiments $H_i(.)$ for corrupted client C :

HYBRID Experiments $H_i^s(.)$

HYBRID^0 : The simulator $\widetilde{\mathcal{S}}$ corresponds to the real world distribution with the actual input $\mathbf{x}$ from client.

HYBRID^1 : Same as Hybrid0, except for a syntactic change. With respect to the layer evaluation by the garbled circuits, we send the one-time pad encryption OTP$(x_{i+1} - [\![r_{i+1}]\!])$by the knowledge of $\mathbf{x}$, $\mathcal{P}_i$ and random tape of the server, instead of the circuits evaluation. Similarly, a share is sent in final round. Thus, when the server adds it with its own share, it gets $x_{i+1} - \langle r_i \rangle$. Hybrid1 is identical to the Hybrid0 with this syntactic change.

HYBRID^2 : The inputs that client provides to each OT execution are changed, in which it acts as the sender. We provide the fake input with '0' to replace the real inputs. This follows the receiver security of obvious transfer protocol, and Hybrid2 is computationally indistinguishable from Hybrid1.

HYBRID^3 : We generate the multiplication triples with the simulator for Beaver's multiplication protocol. With the followed simulation security, the Hybrid3 is computationally indistinguishable from Hybrid2.

HYBRID^4 : We use simulator $\widetilde{\mathcal{S}}$ for the procedure of Beaver's multiplication, with respect to each quadratic approximation layer of the neural network. With the followed simulation security, Hybrid4 is computationally indistinguishable from Hybrid3.

HYBRID^5 : We update the ciphertexts sent by the client in the offline phase. The client sends $\mathsf{Enc}(\mathsf{pk}, 0)$ instead of $\mathsf{Enc}(\mathsf{pk}, r_i)$. The Hybrid5 is computationally indistinguishable from Hybrid4 since this follows the semantic security of the encryption scheme.

HYBRID^6 : We make some changes. With respect to the layer evaluation by the garbled circuits, we send (encryption OTP$[\![r_{i+1}]\!]$) with randomly chosen$[\![r_{i+1}]\!]$ to the server. Similarly, in terms of the each quadratic approximation layer, a share, which is chosen uniformly at random is sent at the final round. Furthermore, a uniformly chosen value$\langle r_1 \rangle$in the offline phase will be sent. Eventually, we note that Hybrid6 is identically distributed to the simulator's $\widetilde{\mathcal{S}}$ output.

Figure 7: Random experiments $H_i(.)$ for corrupted Server $\mathcal{S}$:

$\widetilde{C}$) to the client. $\widetilde{\mathcal{S}}$ runs the corresponding simulator with the Beaver's triples procedure under $\pi_{\texttt{SecureINF}}$.

(3) Similarly, during the online phase: Simulator $\widetilde{\mathcal{S}}$ receives $\mathbf{x} - \langle r_1 \rangle$ from the offline phase, sends $\mathbf{x}$ to the ideal functionality $\mathcal{F}$ and obtains the output y. The simulator $\widetilde{\mathcal{S}}$ performs the corresponding evaluation as:

- Simulator $\widetilde{\mathcal{S}}$ sends the simulated labels for GC layers.
- Simulator $\widetilde{\mathcal{S}}$ evaluates the polynomial approximation layer for Beaver's multiplication procedure. For the output layer, Simulator $\widetilde{\mathcal{S}}$ sends output $\mathbf{y} - [\![r_l]\!]$ to the client.

In this case, a simulator $\widetilde{\mathcal{S}}$ that is given $(C, \mathbf{x})$ can simulate the complete view of C. In Figure 6, we now present that the distribution of real world is computationally indistinguishable to the simulator $\widetilde{\mathcal{S}}$ in the ideal world. We prove this by a sequence of random experiments $H_i(.)$ as shown in Figure 6. It shows that the successive random experiments are computationally indistinguishable. The server's model weights will not be used in the simulator $\widetilde{\mathcal{S}}$ for the final simulated distribution, thus nothing except the prediction results and model architecture will be learned by the corrupted client. This completes the proofs for the case of adversarial client.

Case II: Server $\mathcal{S}$: is corrupted ($\mathcal{S} \in \widetilde{P}_s$). In this case, we assume that the simulator $\widetilde{\mathcal{S}}$ exists as below, once given the inputs $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ...\mathcal{P}_{|N|})$ from the server:

(1) The simulator $\widetilde{\mathcal{S}}$ generates uniform random tape from Servers.

(2) During the offline phase, Simulator $\widetilde{\mathcal{S}}$ sends $\mathsf{Enc}(\mathsf{pk}, 0)$ to the server with chosen pk and receives the ciphertext from the server. And simulator $\widetilde{\mathcal{S}}$ and works as the receiver from server and uses the fake input with "0" as receiver's choice bit. Simulator runs the corresponding simulator $\widetilde{\mathcal{S}}$ for Beaver's triples generating.

(3) On the Online phase, simulator $\widetilde{\mathcal{S}}$ sends $\langle r_1 \rangle$ from the offline phase with an uniformly chosen $\langle r_1 \rangle$. The simulator $\widetilde{\mathcal{S}}$ performs the corresponding evaluation as below:

- Simulator $\widetilde{\mathcal{S}}$ sends the random value back to sever for GC layers.
- $\widetilde{\mathcal{S}}$ uses simulator for Beaver's multiplication to evaluate the polynomial approximation. The random value is sent back to the server at the final round.

We present that the distribution of real world is computationally indistinguishable to the simulated distribution by the following hybrid arguments in the Figure 7. Since the user's input is not used by the simulator in the final simulated distribution, a corrupted server will not know anything in the real world.

Thus, this completes the proof. □

6 Experimental Evaluations

Experimental Environment. Our `Crypto3D` is implemented in Rust, Python, and C++. All the experiments are evaluated on a Ubuntu 20.04.2 LTS server with the NVIDIA RTX A6000.

Datasets and 3D DNN Models. UCF-101 and HMDB-51 human action recognition datasets are utilized to evaluate `Crypto3D`, as shown in Table 2. The UCF-101 dataset consists of 13,320 videos from YouTube, with over 101 categories of human actions. HMDB-51 contains 6,849 video from 51 distinct action classes. The C3D weight model is generated from Sports-1M, which contains more than 1 million YouTube videos annotated with 487 sports classes. The I3D ConvNet model [5] is trained for action recognition with Kinetics-400, which includes 400 different actions. The C3D network helps the temporal information preservation in the first layer and then builds higher level representation of the temporal information with the subsequent layers. The I3D model can further improve C3D via inflating 2D models.

Table 2: Characteristics of video datasets.

	UCF-101	HMDB-51
Average Resolution	360 × 288	360 × 240
Pretrained	Sports-1M	Sports-1M
Clips	13320	6766
Category/Class	101	51
Background	Dynamic	Dynamic
Release Year	2012	2011
Resource	YouTube	Movies, YouTube

Benchmark Systems.[2] We compare the cryptographic inference results and performance of `Crypto3D` with several other systems: Gazelle, Intel SGX (hardware-based protection), 3D secure inference with the MPC protocol (e.g., MP-SPDZ), optimized HE-based privacy-preserving ML frameworks (e.g., CryptoDL, HEANN and E2DM). More details for HE libraries are deferred to the Table 3.

Note that almost all systems only implement and focus on image classification datasets with 2D CNN, which does not directly support the video understanding models. The PPVC [36] proposes a privacy-preserving framework for video classification, but it still utilizes a 2D CNN instead of a 3D ConvNets for the prediction. To fairly compare our `Crypto3D` with them on videos, we redesign and re-implement these systems with non-trivial extensions to enable cryptographic inferences on the 3D models.

Table 3: The detailed description for the benchmarks.

Methods	Description
Gazelle	BFV & lattice encryption library
Intel SGX	Graphene with SGX support
MP-SPDZ	MPC
CryptoDL	SIMD operations Division of ciphertexts is not supported Limited number of computations
HEANN	Scheme with native support for fixed-point approx. arithm.
ED2M	C++ implementation of matrix computation using HEANN

Libraries for FHE. Many existing works use the privacy preserving computation based on the homomorphic encryption (HE). HE enables the computation on the encrypted data without decryption. However, it consists of many restrictions. Therefore, we benchmarks the different state-of-the-art *secure two party inference* frameworks for valuations by integrating different HE libraries. As shown in Table 3, we discuss the details of the provided benchmark. For the

[2] In this work, we focus on comparing the computational costs since the communication overheads of our `Crypto3D` are explicitly lower than existing systems due to the compressed neural networks.

benchmark of Gazelle 3D, we still use Brakerski-Fan-Vercauteren (BFV) scheme from the 2D CNN inference [23]. That supports the integer operations with the Lattice encryption library. And PALISADE is a framework that provides a general API for multiple FHE schemes including BFV, BGV, and CKK.

Microsoft SEAL is a HE library that enable additions and multiplications to be performed on encrypted integers or real numbers. Also, it comes with two different FHE schemes with different properties: BFV and CKK. The modular arithmetic can be performed on encrypted integers by the BFV scheme. And CKKS scheme allows additions and multiplications on encrypted real or complex numbers, however, the approximate results can be generated. The CKKS scheme would be the one of best options for the application such as calculating the total encrypted real numbers or evaluating machine learning models on encrypted data. The BFV scheme is the only option for the application, which requires the exact value.

With respect to the hardware-based protection TEE, the strong privacy and integrity can be guaranteed. In our evaluation, we use the Graphene [49] (a lightweight guest OS) as Intel SGX for the C3D inference execution. It can replaces the Intel SDK for the *enclave* and host process. Furthermore, the MP-SPDZ library is designed for the Secure Multiparty Computation (MPC) implementation. In [36], it uses the MP-SPDZ library for the private video classification based on the Secure MPC. The privacy preserving technique can be achieved and executed for the video classificiation in [36], it still utilizes single frame method for inference with 2D ConvNet instead of the 3D CNN model.

CryptoDL uses the HElib library, it supports the SIMD operations, however there are limitations. Firstly, the division of ciphertexts is not supported. Also, it may causes the incorrect decryption with the exceeded noise, since the additional noise will be added for every computation performed on the ciphertext. Thus, an arbitrary number of computations (i.e., activation functions) can not be supported. In this case, we use the polynomials as activation functions. The fixed point arithmetics can be supported by HEAAN library. This library supports approximate operations between rational numbers. The approximate error depends on some parameters and almost same with floating point operation errors. Cheon et al. [7] used the scheme in this library. And the HEMat is a extension from the HEANN schemes, where it designed for performing a optimized matrix computation with homomorphic encryption.

In summary, we benchmark the following systems based on the C3D model: Gazelle (3D), Intel SGX (3D), MP-SPDZ (3D), CryptoDL (3D), HEANN (3D), E2DM (3D), CryptFlow2(3D), and Cheetah(3D). However, GALA cannot be extended due to its end-to-end 2D structure or lack of source code.

6.1 The Performance of `Crypto3D`

Comparison on UCF-101 and HMDB-51 with C3D and I3D. For the generic human action recognition setting, we work on the two most representative datasets (UCF-101 and HMDB-51). The pre-trained weight models we used are extracted from Sports-1M for C3D and kinetics-400 for I3D, respectively.

Figure 8 shows faster amortized execution time for the HMDB-51 than UCF-101 in both C3D and I3D compared with benchmarks. C3D performs better than I3D on both UCF-101 and HMDB-51 as well. Compared to the pre-trained model and DNNs architecture, we find that the dataset would not be the main factor for the performance impact under this case. Note that a further comparison will be conducted after the network pruning process on C3D. Further information can be found in the Table 4.

Weight-pruning Optimization on SLR Approach. In order to reduce the computation cost and model size, we use SLR [14] based weight-pruning for C3D pre-trained model. We have three pruning models which are presented with different pruning parameters (0.95, 0.9, and 0.5) in Table 5. The best performance model (Model 0.5) is used for further comparison with SOTA secure systems.

6.2 Further Comparison with SOTA Systems

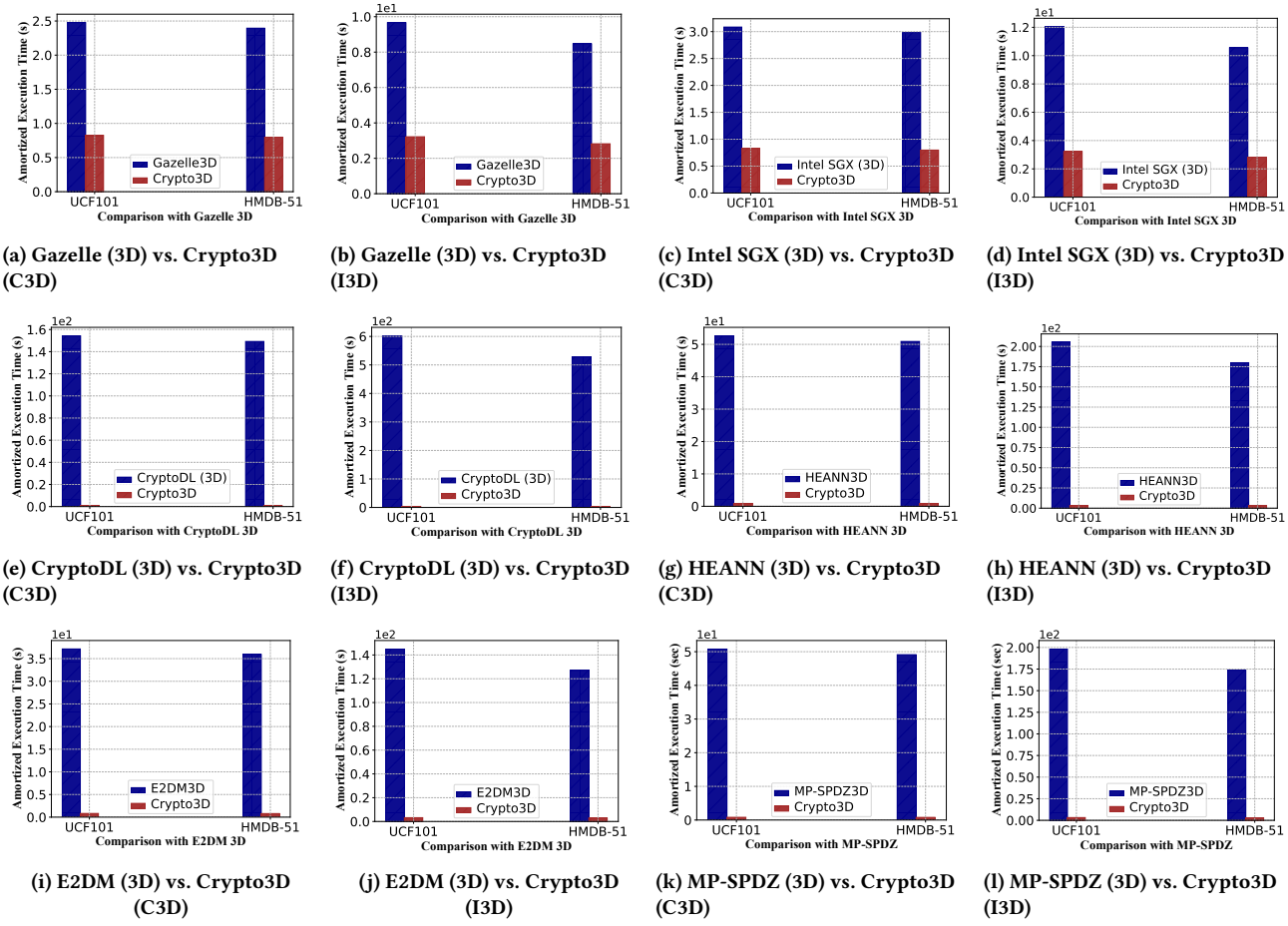
Runtime Comparison with the SOTA Systems. Table 4 summarizes the methodology, library, total execution time, speedup, and amortized time of secure inference for an input video from the UCF-101 dataset among the state-of-the-art secure systems. `Crypto3D` outperforms all other 3D frameworks significantly. Figure 9a presents the times of speedup with prior secure systems. The execution time of `Crypto3D` is over 554.68×, 189.21×, 182.61×, 133.56×, 11.09×, 8.90×, 3.71×, 12.97×, and 1.49× faster than CryptoDL (3D), HEANN (3D), MP-SPDZ (3D), E2DM (3D), Intel SGX (3D), Gazelle (3D), Delphi (3D), CryptFlow2 (3D), and Cheetah (3D), respectively. These results show that `Crypto3D` (optimized) is much more efficient in 3D privacy-preserving video input inference. Additionally, `Crypto3D` only takes an average of 0.28 sec (before 0.83 sec) to process the secure inference for each frame, while other HE-based frameworks take much longer time.

Comparison with PPVC [36] (low accuracy). Additionally, we compare with PPVC [36], the SOTA method for secure two-party video classification, which uses MPC for private classification. Table 4 shows that PPVC takes slightly less time since it utilizes a 2D ConvNet-trained model for video classification instead of the C3D model (*however, this method can only obtain a very low accuracy of 56%*, as shown in Table 6). The architecture of ConvNets is designed as [(CONV-RELU)-POOL]-[(CONV-RELU)*2-POOL]*2- [FC-RELU]*2-[FC-SOFTMAX] on the FER 2013 dataset. Unlike 2D ConvNets, the C3D architecture employs 3D convolutional operations, such as Conv3D, to extract spatial-temporal features across multiple consecutive frames. Moreover, the C3D model [48] is trained on the Sports-1M dataset [24] for action recognition, while PPVC is trained on the RAVDESS dataset [31] for emotion detection.

Layer Evaluation. Figure 9b shows the execution time on the convolution layers in `Crypto3D` when running on the GPU using different batch sizes $b \in \{1, 5, 10\}$. Given different batch sizes, the execution time on each convolutional layer is distinct. We observe that the results of batch size 1 outperform the results of using other batch sizes. Similar to the Delphi, the prime field we used enables the implementation for GPU libraries for the linear operations. By amortizing the batch convolutions over different inputs together, we can reduce the cost compared to single convolutions.

6.3 Accuracy

In our work, the video classifier samples every 15th frame, classifies it with the above ConvNet, and assigns the final class label as the label that has the highest average probability across all frames in



(a) Gazelle (3D) vs. Crypto3D (C3D)

(b) Gazelle (3D) vs. Crypto3D (I3D)

(c) Intel SGX (3D) vs. Crypto3D (C3D)

(d) Intel SGX (3D) vs. Crypto3D (I3D)

(e) CryptoDL (3D) vs. Crypto3D (C3D)

(f) CryptoDL (3D) vs. Crypto3D (I3D)

(g) HEANN (3D) vs. Crypto3D (C3D)

(h) HEANN (3D) vs. Crypto3D (I3D)

(i) E2DM (3D) vs. Crypto3D (C3D)

(j) E2DM (3D) vs. Crypto3D (I3D)

(k) MP-SPDZ (3D) vs. Crypto3D (C3D)

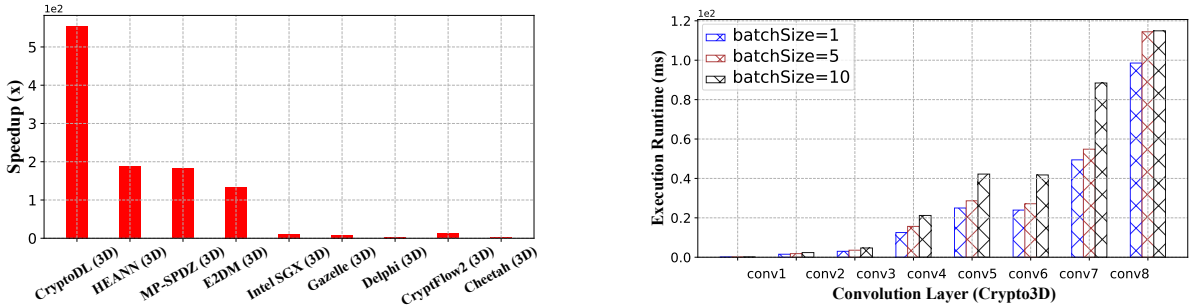
(l) MP-SPDZ (3D) vs. Crypto3D (I3D)

Figure 8: `Crypto3D` (ours) vs. SOTA systems (Gazelle (3D), Intel SGX (3D), CryptoDL (3D), HEANN (3D), E2DM (3D)) based on C3D and I3D models on UCF-101 and HMDB-51 datasets, respectively.

Table 4: Comparison with SOTA secure systems, their features, and performance (*non-trivial extensions from 2D to 3D*) on one input video, RGB tensors of size 16×112×112. `Crypto3D` is significantly more efficient than other systems. The execution time of `Crypto3D` is over 554.68×, 189.21×, 182.61×, 133.56×, 11.09×, 8.90×, 3.71×, 12.97×, and 1.49× faster than CryptoDL (3D), HEANN (3D), MP-SPDZ (3D), E2DM (3D), Intel SGX (3D), Gazelle (3D), Delphi (3D), CryptFlow2 (3D), and Cheetah (3D), respectively.

Method (3D)	Design	Library	Network	Optimization	Runtime (Sec)	Amortized (Sec)
Gazelle	HE, GC, SS	PALISADE	C3D	N/A	1916.48	2.48
Intel SGX	TEE	-	C3D	N/A	2387.77	3.08
PPVC [36]	MPC, SS	MP-SPDZ	2D CNN	N/A	511.64[36]	-
MP-SPDZ	MPC, SS	MP-SPDZ	C3D	N/A	39303.72	50.78
CryptoDL	HE	HELIB	C3D	N/A	119388.28	154.25
HEANN	HE	HEANN	C3D	N/A	40725.29	52.62
E2DM	HE	HEANN	C3D	N/A	28747.26	37.14
Delphi	HE, GC, SS	SEAL	C3D	**Neural Arch. Search**	798.54	1.03
CrypTFlow2	HE, SS	SEAL	C3D	N/A	2790.79	3.6
Cheetah	HE, SS	SEAL	C3D	N/A	318.30	0.41
`Crypto3D`	HE, GC, SS	SEAL	C3D	**Irregular Pruning**	215.24	0.28

the video. The accuracy of video classification can vary depending on the training model used. Delphi enhances Gazelle by integrating



(a) Comparison with SOTA cryptographic systems, in terms of Speedup (x) over them by `Crypto3D` (optimized).

(b) Execution runtime vs. convolutional layers and different batch sizes (batch size varies as 1, 5 and 10).

Figure 9: `Crypto3D` optimization

Table 5: Weight-pruning optimization (based on SLR schemes) comparison on C3D pre-trained model processing with different sparsity (0.95, 0.9 and 0.5) on layer weight. Model 0.5 is used as `Crypto3D` (optimized) for further comparison with SOTA systems in Table 4.

	Sparsity (0.95)	Sparsity (0.9)	Sparsity (0.5)*
Compression Rate:	18.16	9.541	1.99
(#) of Nonzero Params:	4317867	8216849	39408704
Params Pruned (%):	95	90	50

Table 6: Inference accuracy comparison between PPVC, Delphi, and `Crypto3D`. By employing a distinct optimization technique, `Crypto3D` attains the same level of accuracy as Delphi but with a significantly reduced runtime.

Method	Network	Optimization	Accuracy	Runtime (Sec)
PPVC	2D Conv	N/A	56%	511.64
Delphi (3D)	C3D	Neural Arch. Search	82.3%	798.54
`Crypto3D`	C3D	Irregular Pruning	82.4%	215.24

garbled circuits and quadratic polynomials for activation functions, yielding results that closely match the original ones. As mentioned earlier, our approach surpasses existing systems in both performance and accuracy, achieving high performance without sacrificing accuracy. To facilitate further fair comparisons in optimization, we strive to identify an appropriate placement or network configuration for Delphi (3D) through optimization. We utilize NAS to capitalize on performance-accuracy tradeoffs to ensure that accuracy above a specified threshold (82.3%), as shown in Table 6 with a margin of 0.1% lower than ours. Similarly, Cheetah and CrypTFlow2, when based on our environment, exhibit an accuracy lower than 70% without optimization, even though they demonstrate high performance with effective approximate truncation. PPVC inherits the MP-SPDZ and Gazelle techniques (MPC), resulting in an improvement of 0.8%. Due to the hardware security technique, the accuracy will not be impacted in Intel SGX. Also, CryptoDL, HEANN, and ED2M are all mainly based on the HE with different libraries and settings parameters. The results are close to the original accuracy in the test set (< 70%). To facilitate comparison with our work, we modify the network structures and retrain them using special LHE-friendly non-linear functions. Additionally, we make adjustments to the structure, approximate function/softmax, and selected layers in the UCF-101 dataset to better align with the frameworks. However, such alterations may potentially have a negative impact on the accuracy of compressed models in order to satisfy high performance. Nonetheless, in our model, we effectively manage the trade-off between accuracy with explicitly lower communication overheads and compression rate, achieving 82.4% accuracy while ensuring privacy.

7 Conclusion

Many existing techniques are proposed to perform the *secure two-party inferences* with the cryptographic schemes for the deep neural networks. However, they cannot be directly applied to video inferences which extracts spatio-temporal (3D) features for more accurate video recognition. We propose `crypto3D`, the first cryptographic inference technique based on spatial-temporal (3D) features, which (i) privately infers videos with the C3D and I3D DNN models; (ii) optimizes the matrix operations and ciphertext packing technique to boost efficiency; (iii) adopts weight pruning optimization for further boosting the efficiency of cryptographic C3D and I3D. `Crypto3D` is significantly more efficient than SOTA cryptographic inference systems, and it can also achieve 82.4% accuracy on private inferring videos, which is also significantly more accurate than SOTA cryptographic inference systems.

Acknowledgments

We sincerely thank the anonymous reviewers for their constructive comments and suggestions. This work is supported in part by the National Science Foundation (NSF) under Grants No. CNS-2308730, CNS-2302689, CNS-2432533, CNS-2319277, and CMMI-2326341. It is also partially supported by the Wentworth Faculty Awarded Summer 2024 Bistline Grants. Meanwhile, thanks to the Institute for Experiential AI - Northeastern University for their generous support and the use of their facilities, which contributed to the progress of this work.

References

[1] Marshall Ball, Brent Carmer, Tal Malkin, Mike Rosulek, and Nichole Schimanski. Garbled Neural Networks are Practical. *IACR Cryptol. ePrint Arch.*, 2019.

[2] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In ACM CCS, 2017.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning* , 1–122.

[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory* 6, 3 (2014), 13:1–13:36.

[5] João Carreira and Andrew Zisserman. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *IEEE CVPR 2017* .

[6] Xuhui Chen, Jinlong Ji, Lixing Yu, Changqing Luo, and Pan Li. 2018. SecureNets: Secure Inference of Deep Neural Networks on an Untrusted Cloud. In *Proc. of the ACML, 2018*, Vol. 95, 646–661.

[7] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology*, 2017, 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Vol. 10624.

[8] Minsu Cho, Ameya Joshi, Brandon Reagen, Siddharth Garg, and Chinmay Hegde. Selective Network Linearization for Efficient Private Inference. In ICML 2022.

[9] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. 2012. Multiparty Computation from Somewhat Homomorphic Encryption. CRYPTO'12.

[10] Shuya Feng, Meisam Mohammady, Han Wang, Xiaochen Li, Zhan Qin, and Yuan Hong. 2024. DPI: Ensuring Strict Differential Privacy for Infinite Data Streaming. In *IEEE Symposium on Security and Privacy*, 2024.

[11] Taher El Gamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* 31, 4 (1985), 469–472.

[12] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In ICML 2016,

[13] Deniz Gurevin, Mikhail A. Bragin, Caiwen Ding, Shanglin Zhou, Lynn Pepin, Bingbing Li, and Fei Miao. 2021. Enabling Retrain-free Deep Neural Network Pruning Using Surrogate Lagrangian Relaxation. In IJCAI, 2021.

[14] Deniz Gurevin, Shanglin Zhou, Lynn Pepin, Bingbing Li, Mikhail Bragin, Caiwen Ding, and Fei Miao. 2020. Enabling retrain-free deep neural network pruning using surrogate lagrangian relaxation. (2020).

[15] Joseph Halpern and Vanessa Teague. 2004. Rational secret sharing and multiparty computation. In *Proc. of the thirty-sixth annual ACM STOC*. 623–632.

[16] Lucjan Hanzlik, Yang Zhang, Kathrin Grosse, Ahmed Salem, Maximilian Augustin, Michael Backes, and Mario Fritz. 2021. MLCapsule: Guarded Offline Deployment of Machine Learning as a Service. In *IEEE Conference on CVPR Workshops 2021* .

[17] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N. Wright. 2018. Privacy-preserving Machine Learning as a Service. PoPETs, 2018.

[18] Yuan Hong, Jaideep Vaidya, Haibing Lu, Panagiotis Karras, and Sanjay Goel. 2022. Collaborative Search Log Sanitization: Toward Differential Privacy and Boosted Utility. In *IEEE Transactions on Dependable and Secure Computing*. 2015.

[19] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference. In *USENIX'22*.

[20] Nandan Kumar Jha, Zahra Ghodsi, Siddharth Garg, and Brandon Reagen. 2021. DeepReDuce: ReLU Reduction for Fast Private Inference. In *ICML 2021,*

[21] Nandan Kumar Jha and Brandon Reagen. 2023. DeepReShape: Redesigning Neural Networks for Efficient Private Inference. *CoRR* abs/2304.10593 (2023).

[22] Xiaoqian Jiang, Miran Kim, Kristin E. Lauter, and Yongsoo Song. 2018. Secure Outsourced Matrix Computation and Application to Neural Networks. In *CCS*.

[23] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In *USENIX Security*, 2018, 1651–1669.

[24] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In *IEEE Conference on CVPR 2014*.

[25] Julien Keuffer, Refik Molva, and Hervé Chabanne. 2018. Efficient Proof Composition for Verifiable Computation. In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proc., Part I (Lecture Notes in Computer Science, Vol. 11098)*, 152–171.

[26] Brian Knott, Shobha Venkataraman, Awni Y. Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. 2021. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In *Advances in NeurIPS 2021*.

[27] Hildegard Kuehne, Hueihan Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre. 2011. HMDB: A large video database for human motion recognition. In *ICCV, 2011*.

[28] Souvik Kundu, Shunlin Lu, Yuke Zhang, Jacqueline Tiffany Liu, and Peter A. Beerel. 2023. Learning to Linearize Deep Neural Networks for Secure and Efficient Private Inference. In *ICLR, 2023*.

[29] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. 2017. Oblivious Neural Network Predictions via MiniONN Transformations. In *ACM CCS*, 2017 .

[30] Bingyu Liu, Shangyu Xie, and Yuan Hong. 2020. PANDA: Privacy-Aware Double Auction for Divisible Resources without a Mediator. In *AAMAS*, 2020 .

[31] Steven R Livingstone and Frank A Russo. 2018. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PloS one* 13, 5 (2018).

[32] Meisam Mohammady, Lingyu Wang, Yuan Hong, Habib Louafi, Makan Pourzandi, and Mourad Debbabi. Preserving both privacy and utility in network trace anonymization. In *CCS*, 2018.

[33] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A Cryptographic Inference Service for Neural Networks. In *USENIX Security*, 2020.

[34] Payman Mohassel and Peter Rindal. 2018. ABY^3: A Mixed Protocol Framework for Machine Learning. In *ACM CCS, 2018*.

[35] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques*.

[36] Sikha Pentyala, Rafael Dowsley, and Martine De Cock. 2021. Privacy-Preserving Video Classification with Convolutional Neural Networks. In *ICML 2021*.

[37] NhatHai Phan, Xintao Wu, Han Hu, and Dejing Dou. Adaptive Laplace Mechanism: Differential Privacy Preservation in Deep Learning. In *IEEE ICDM, 2017*.

[38] Rishabh Poddar, Ganesh Ananthanarayanan, Srinath Setty, Stavros Volos, and Raluca Ada Popa. 2020. Visor: Privacy-Preserving Video Analytics as a Cloud Service. In *USENIX Security, 2020.* , pp. 1039–1056.

[39] Michael O. Rabin. 2005. How To Exchange Secrets with Oblivious Transfer. *IACR Cryptol. ePrint Arch.* (2005), 187.

[40] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. CrypTFlow2: Practical 2-Party Secure Inference. In *ACM CCS*, 2020,

[41] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E. Lauter, and Farinaz Koushanfar. XONN: XNOR-based Oblivious Deep Neural Network Inference. In *USENIX Security, 2019*, 1501–1518.

[42] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. In *AsiaCCS, 2018*.

[43] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. 2018. Deepsecure: scalable provably-secure deep learning. In *Proc. of the 55th Annual DAC, 2018*

[44] Reza Shokri and Vitaly Shmatikov. Privacy-Preserving Deep Learning. In CCS.

[45] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *CoRR* abs/1212.0402 (2012).

[46] Shruti Tople, Karan Grover, Shweta Shinde, Ranjita Bhagwan, and Ramachandran Ramjee. Privado: Practical and Secure DNN Inference. *CoRR* abs/1810.00602 (18).

[47] Florian Tramèr and Dan Boneh. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. In *ICLR 2019,* .

[48] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In *IEEE ICCV, 2015*. 4489–4497.

[49] Chia-che Tsai, Kumar Saurabh Arora, Nehal Bandi, Bhushan Jain, William Jannen, Jitin John, Harry A. Kalodner, Vrushali Kulkarni, Daniela A. S. de Oliveira, and Donald E. Porter. 2014. Cooperation and security isolation of library OSes for multi-process applications. In *EuroSys*. 9:1–9:14.

[50] Di Wang, Minwei Ye, and Jinhui Xu. 2018. Differentially Private Empirical Risk Minimization Revisited: Faster and More General. *CoRR* abs/1802.05251 (2018).

[51] Han Wang, Shangyu Xie, and Yuan Hong. 2020. VideoDP: A Flexible Platform for Video Analytics with Differential Privacy. PoPETs, 2020.

[52] Han Wang, Yuan Hong, Yu Kong, and Jaideep Vaidya. 2020. Publishing Video Data with Indistinguishable Objects. *EDBT*, 2020,.

[53] Shangyu Xie, Han Wang, Yu Kong, and Yuan Hong. 2022. Universal 3-Dimensional Perturbations for Black-Box Attacks on Video Recognition Systems. In *IEEE Symposium on Security and Privacy*, 2022.

[54] Shangyu Xie, Yan Yan, and Yuan Hong. 2024. Stealthy 3D Poisoning Attack on Video Recognition Models. In *TDSC*, 2024.

[55] Masashi Yamane and Keiichi Iwamura. 2020. Secure and Efficient Outsourcing of Matrix Multiplication based on Secret Sharing Scheme using only One Server. In *IEEE CCNC, 2020,*.

[56] Andrew Chi-Chih Yao. 1986. How to Generate and Exchange Secrets (Extended Abstract). In IEEE *FOCS*, 1986, 162–167.

[57] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially Private Model Publishing for Deep Learning. In *IEEE Symposium on Security and Privacy*, 2019. 332–349.

[58] Qiao Zhang, Chunsheng Xin, and Hongyi Wu. 2021. GALA: Greedy ComputAtion for Linear Algebra in Privacy-Preserved Neural Networks. In *NDSS, 2021*.

[59] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. 2018. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *ECCV*, 2018, pp. 184–199.

[60] Hongchao Zhou and Gregory W. Wornell. Efficient homomorphic encryption on integer vectors and its applications. In *ITA, 2014*.

[61] Shanglin Zhou, Mimi Xie, Yufang Jin, Fei Miao, and Caiwen Ding. An end-to-end multi-task object detection using embedded gpu in autonomous driving. In *IEEE ISQED*, 2021, 122–128.