Privacy-Preserving Video Classification with Convolutional Neural Networks

Sikha Pentyala

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science and Systems

University of Washington

2020

Reading Committee:

Martine De Cock, Chair

Anderson Nascimento

Rafael Dowsley

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington

Abstract

Privacy-Preserving Video Classification with
Convolutional Neural Networks

Sikha Pentyala

Chair of the Supervisory Committee:
Professor Martine De Cock
School of Engineering and Technology

Video classification using deep learning is widely used in many applications such as facial recognition, gesture analysis, activity recognition, emotion analysis, etc. Many applications capture user videos for classification purposes. This raises concerns regarding privacy and potential misuse of videos. As videos are available on the internet or with the service providers, it is possible to misuse them such as to generate fake videos or to mine information from the videos that goes beyond the professed scope of the original application or service. The service provider on the other hand typically cannot provide the trained video classification model to be run on the client's side either, due to resource constraints, proprietary concerns and the risk for adversarial attacks. There is a need for technology to perform video classification in a privacy-preserving manner, i.e. such that the client does not have to share their videos with anyone without encryption, and the service provider does not have to show their model parameters in plaintext. In this thesis, we propose privacy-preserving single frame based video classification with a pre-trained convolutional neural network based on Secure MultiParty Computation. The pipeline for securely classifying a video involves three major protocols: oblivious selection of frames in a video, securely classifying individual frames in the video using existing protocols for image classification, and secure label aggregation across frames to obtain a single class label for the video. We perform run-time experiments for the use

case of emotion detection in a video to demonstrate the feasibility of our proposed methods in practice.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my advisor, Dr. Martine De Cock, for the continuous support both academically and personally during my Masters' study and research, for her patience, motivation, encouragement, and technical guidance and insights. I would also like to thank Mr. Kyle Bittner for directing us to the MP-SPDZ software. Additionally, I wish to thank Dr. Rafael Dowsley from Monash University and Dr. Anderson C.A. Nascimento from University of Washington for their insightful comments, especially in the area of cryptography. Other thanks go to the students and co-workers in the group who have always been open to discussions. I would like to thank Mr. Marcel Keller for making the MP-SPDZ framework available, and for his assistance in the use of the framework. Additional mention to Mr. Anders Dalskov and Mr. Daniel Escudero for providing initial works for secure building blocks of convolutional neural networks for ring algebraic structures.

DEDICATION

to my husband and my daughter

Chapter 1

INTRODUCTION

Artificial neural networks (ANNs) that use deep learning architectures such as Convolutional Neural Networks (CNNs) have been established as a powerful class of machine learning models for image processing. ANN architectures are being popularly used for sequential data such as text and speech too. One of the other popular data domains is videos. Videos can be considered as stacks of images where subsequent images in the video are correlated with respect to their semantic contents and temporal characteristics, making them sequential in nature. We can use ANNs for various applications on videos. Video classification is one such application where, given a set of categories or classes, one needs to analyze the contents of the video to categorize it to a particular class.

Video classification performed using such ANNs and deep learning is used in various applications, such as facial recognition [58], gesture analysis [48], activity recognition [55, 60, 87], motion analysis [3], micro-expression analysis[1], behavioral analysis [56, 78], eye gaze estimation[2], and emotion and sentiment recognition. The above mentioned applications have great potential to improve the productivity and quality of human life. Empathy-based AI systems that detect when someone is in emotional distress, learning assistants that detect how the student is feeling, and personal digital assistants that detect the user's mood and make suggestions to help improve it, are nice illustrations of this. Detection of driver drowsiness, baby monitoring systems, home security systems, activity recognition in care centers such as child care centers, old age centers, health care centers, detection of abusive activities, eye gaze estimation, and detecting concentration of students in online courses [70] are other

[1]The Third Micro-Expression Grand Challenge (MEGC) Workshop in 15th IEEE international conference on automatic face and gesture recognition, 2020

[2]https://eyeware.tech/

potential useful applications for video classification.

While there are many benefits for video classification applications, many of the most powerful existing and future applications of Artificial Intelligence (AI) rely on personal data. The above mentioned applications of video classification require the ANNs to be trained on very large amounts of data. Such trained models are generally available to some specific organizations or with commercial service providers. We hereafter refer them as *owners of the video classifier*. When a video needs to be classified, the video is given to these owners and they provide the classification of the video. Providing videos in this way may lead to invasion of privacy of the *video owner*. It may so happen that these videos remain on the Internet, even after classification, and become available for potential misuse, such as generation of deepfakes, or for scraping as done for instance by Clearview AI [31]. The video owner may not even have control to delete such videos.

A seemingly straightforward technique to keep the users' videos private is by deploying the video classifier at the users' end. But such a scenario may compromise the privacy of the trained video classifier. It may be that the owners of the proprietary video classifier want to conceal their model from competitors. It can be also be that owners of the video classifier are concerned that an adversary with access to the classifier might find ways to deliberately make the model classify wrongly through evasion attacks, such as in security applications with facial recognition where security can be compromised by deepfake generation. Furthermore, it is well known that powerful deep learning models can memorize their training examples, which the owners of the video classifier would not want to expose by making the model available. Moreover the video owner may not have enough computational resources to perform the task of video classification. Considering the computationally intensive task of video classification, it is possible that both the video owner and the video classifier outsource the task to a set of servers in the cloud, popularly known as MLaaS (Machine Learning as a Service) providers. Though this addresses the problem due to limited resources, the problem of keeping both the video and the video classifier private and secure still remains, as an adversary can corrupt any of the servers to leak information.

There arises a need to protect the privacy of the individuals [77] – in this context the owner of the video as well as the video classifier. Concerns regarding privacy of user data are giving rise to new laws and regulations such as the European GDPR and the California Consumer Privacy Act (CCPA), as well as a perceived tension between the desire to protect data privacy on one hand, and to promote an economy based on free-flowing data on the other hand [41]. The E.U. is for instance considering a three-to-five-year moratorium on face recognition in public places,given its significant potential for misuse [19].

We propose to solve this problem of classifying a video by keeping both the video and the video classifier private so that one can still take advantage of the impactful applications of video classification. Privacy-preserving machine learning (PPML) has been receiving a lot of attention in handling such sensitive data [68, 90] while utilizing the power of machine learning algorithms. Much progress has been made in the area of PPML using Secure Multi-Party Computation (MPC) [22], an umbrella term for cryptographic approaches that allow two or more parties to jointly compute a specified output from their private information in a distributed fashion, without revealing the private information to each other. Many solutions have been proposed for privacy preserving inference using deep learning models in images [2, 25, 40, 47, 63, 73, 74, 75] and audio classification [8]. We build upon these existing works and propose an end-to-end secure video classification solution using a single-frame based approach (cfr. Chapter 4).

Our solution is summarized in Figure 1.1. We assume that *Alice* is the owner of the video that needs to be classified and *Bob* has the trained video classifier. Both of them outsource the task of video classification to an MLaaS provider consisting of a set of untrusted servers S_1, S_2, S_3 collectively called as S. In step 1, *Alice* and *Bob* "secret share" their private data (cfr. Chapter 3) and distribute the shares to S in a way that no server alone can infer any information about the private data. In step 2, the servers perform joint computations on these secret shares using MPC protocols such that no private information is leaked and at each computation the servers hold the secret shares of the computed value (cfr. Chapter 4). S performs computations to classify a video in 3 major steps:

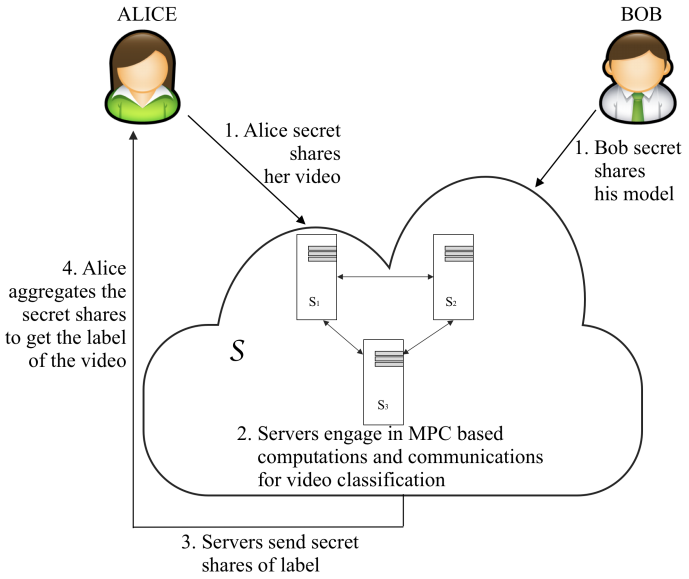

Figure 1.1: Privacy-preserving video classification as an outsourced computation problem to 3 parties (servers)

- Selecting video frames in an oblivious manner.
- Secure frame inference leveraging existing work on image classification.
- Privacy-preserving label aggregation across the frames.

At the end of step 2, the servers will hold secret shares of the class label of the video. Our thesis focuses on building a solution for step 2. The servers, in step 3, send the secret shares of the class label of the video to *Alice*, who aggregates the secret shares to construct the class label of the video in step 4.

MPC aims at providing security against corrupted parties (the servers in Figure 1.1) thus ensuring privacy of the data and correct computations. Various MPC schemes and protocols exist to prevent different types of attacks by corrupted parties or provide security at different levels corresponding to different threat models. One of the factors considered while designing these MPC schemes and protocols is the number of parties involved in performing joint computations and the number of parties that can get corrupted. Some of the efficient

protocols are available for three party computation (3PC), i.e a setup with total of three parties of which one can get corrupted. In our work, we consider this *3PC honest majority setting* which is growing popular in PPML literature [25, 47, 74, 82], i.e we have 3 servers (S_1, S_2, S_3) who will engage in performing joint computations and corruption of only one server can be tolerated. Our work considers providing security against two types of corruptions: *passive* where a corrupted party will abide by the protocols but tries to gather some information on others' private data; and *malicious* where a corrupted party deviates from protocols and can cause incorrect computations. The MPC protocols that we design can also be used in other security settings by changing the underlying schemes. For example, by changing the underlying MPC schemes and protocols, a change in setup from 3PC to 2PC (with dishonest majority setting as well) can be done but with an increase in runtime. To elaborate, in such a 2PC setup for our scenario, only *Alice* and *Bob* get involved in the joint computations and do not outsource the computation to a set of servers. In a 2PC setup with dishonest majority setting, *Alice* and *Bob* can only trust oneself. In other words, a 2PC setup with a dishonest majority setting is where the number of corrupted parties can be greater than or equal to half the number of total parties, i.e, with total of two parties each party can trust only itself, as either *Alice* or *Bob* or both can get corrupted leading to total number of corrupted parties greater than or equal to half.

We start with a discussion of related work in Chapter 2 and then provide preliminaries on MPC and on video classification with CNNs in Chapter 3. Our solution for secure video classification uses the MPC protocols that we propose in Chapter 4. We demonstrate the feasibility of performing secure video classification by evaluating our proposed approach with no information leakage in an application for human emotion detection from video on the benchmark RAVDESS dataset in Chapter 5. Our solution achieves accuracy at par with those in the literature for this dataset. Our experiments show that a video in a 3PC, honest majority setting can be classified securely in 13.8 seconds with passive security and 82.7 seconds with active security on AWS c5.4xlarge instances and it can be made more efficient with high-end computational resources for secure video classification in near-real time. This

shows that secure video classification based on MPC is feasible in practice.

Chapter 2
RELATED WORK

2.1 Secure image classification

Privacy-preserving inference has received considerable attention over the past few years. Many notable works with MPC-based approaches and frameworks for secure image classification are available in the literature [2, 8, 11, 25, 40, 46, 47, 63, 69, 18, 73, 74, 75, 82, 83]. Many works also focus on secure outsourced computations in the setting of popular MLaaS [24, 25, 42, 69, 18, 82]. We incorporate some of the existing work for secure classification of images [25] in our single-frame approach to securely classify individual frames in the video whilst developing an end-to-end MPC-based video classification solution in the context of outsourced MLaaS models.

2.2 Secure video classification

The area of performing video classification securely is being explored in many works. *Non-cryptographic techniques* like anonymizing faces in videos [72], pixel randomization to hide the user's identity [34], compressing video frames to achieve visual shielding effect [54], lowering resolution of videos [76], using autoencoders to maintain privacy of the user's data [27], and changes in ways the videos are captured [85] do not provide any formal privacy guarantees and may also affect the accuracy of the inference made. Works using differential privacy (DP) [84] for video classification have been proposed for learning. Differential privacy introduces noise or generates new data based on the original data at the user end and the inference is then provided on new data. This affects accuracy of the inference being made and it is possible that information is leaked, especially in case of single inference. Recent solutions also include using secure hardware (trusted execution environments or TTEs) for

privacy-preserving video analytics [71].

To have complete privacy of the data and ensure no leakage of private information without use of TTEs, we focus on *cryptographic methods*. To the best of our knowledge, the only existing work on cryptography based video classification is based on Homomorphic Encryption (HE) [92]. Under an HE scheme the end user performs encryption using $E(.)$ on the data (or the video) x to be labeled and sends it to the owner of a video classifier who performs inference using $f(.)$. $f(.)$ consists of operations compatible with HE scheme being used. The inference $y = f(x)$ is instead translated to $E(y) = f(E(x))$ where y is the class label in clear text. The enrcypted inference $E(y)$ is then sent to the end user who performs decryption using $D(.)$ to obatin $y = D(E(x))$. While HE-based solution provides higher security than the above mentioned techniques and lower communication costs than the MPC-based solution, it has higher computational costs as all the operations are performed on the encrypted data. This makes it computationally expensive and not a viable solution for practical video classification.

2.3 Emotion detection in videos

Emotion recognition has been a significant research topic because of its potential use in emerging empathy-based AI systems. Emotions have strong influence in applications that require attention detection, action recognition, behavior analysis, motivation detection for decision-making, detecting whether the user is motivated, frustrated, curious or under stress for learning based applications. There has been a notable amount of work in the literature for emotion recognition using various modalities and features [7, 37, 39, 64, 86], including videos [1, 10, 97, 33, 59, 64, 65, 26]. A few works are available that infer emotions on video modality using the RAVDESS dataset that we use in this thesis. In the existing work, authors use various artificial neural network architectures such as synchronous graphs neural networks [59], stacked autoencoders [6], RNN [10] and ConvNet-LSTM achitectures [1], reporting accuracies in the 57%-82% range, depending on the number of emotion classes included in the study (6 to 8). Jaiswal and Provost [36] have studied privacy metrics and

leakages when inferring emotions from data. To the best of our knowledge, a solution using MPC for privacy-preserving emotion recognition in video, as we propose in this thesis, has not been explored yet.

Chapter 3

PRELIMINARIES

3.1 Secure Multi-Party Computation

Secure multi-party computation (SMC/MPC) is an umbrella term for cryptographic methods that allow two or more parties to jointly perform computations to calculate the output of a previously agreed function on their combined inputs, without revealing the private information. In other words, SMC methods reveal nothing beyond what is revealed by the output and the inputs in possession of the adversary. Let us assume parties $P_1, P_2 \dots P_n$ respectively hold the inputs $x_1, x_2 \dots x_n$ and want to compute $y = f(x_1, x_2 \dots x_n)$ without any party, P_i, revealing its private data – also called secret – x_i to any other party. None of the above parties trust any other party outside or inside their group. This thus removes a scenario where the parties can assume an ideal trusted party exists to whom they can provide their private information x_i and get back the final value y. SMC methods allow to securely compute y by protecting the private data x_i of each party P_i without disclosing it to other parties within the group or outside the group (the communications happen only among the parties and a secure, reliable communication channel is assumed), thus focusing on the corruption of the parties inside the group by an adversary $\mathcal{A}$.

Secret Sharing To jointly calculate y correctly and securely, i.e. without revealing x_i to any party within or outside the group, parties can split their secret based on some randomness and distribute it to other parties such that every party has a share of the secret. This secret can be reconstructed only when a sufficient number of shares and parties, defined based on the scheme being used, are available. These individual shares of the secret provide no information about the original secret. If n parties are involved and at the least $t + 1$ parties are required to reconstruct the secret then it is generally termed as (t, n) threshold secret

sharing scheme. Using the secret shares the intended function can be computed. Let us say P_1 wants to secret share its secret x_1 to P_2 and P_3, then based on the secret sharing scheme used, x_1 will be split randomly into secret shares x_{11}, x_{12} and x_{13} and distributed accordingly. One of the simplest distribution schemes can be where P_1 will hold x_{11}, P_2 will hold x_{12} and P_3 will hold x_{13}. We denote the secret shares of x as $[\![x]\!]$.

Linear secret sharing schemes Linear secret sharing schemes (LSSS) imply that any linear operation performed on the secret shares will be reflected on the reconstructed share. This form of homomorphism enables the parties to compute a function $f(.)$ on the secret indirectly, by individually and locally computing $f(.)$ on the secret shares without need for any communication. In other words, the secret shares that the parties hold (either after distribution or after performing local computations) are some fixed linear combination of the secret. For example to calculate, $y = x_1 + x_2$ where x_i is secret of P_i, assume that the secrets are randomly and uniformly split as $x_1 = x_{11} + x_{12}, x_2 = x_{21} + x_{22}$. x_{ij} are the secret shares of the secret x_i owned by P_i who also holds all the secret shares with $j = i$. After distribution using the above said scheme, P_1 holds x_{11}, x_{21} and P_2 holds x_{21}, x_{22}. Each party performs local additions on their secret shares resulting in secret shares y_1, y_2 of the sum, i.e. $y = y_1 + y_2$ where $y_1 = x_{11} + x_{21}$ and $y_2 = x_{12} + x_{22}$. In other words, after performing the local additions, the parties hold $[\![y]\!]$. The secret shares of the sum could then be combined to reveal the final secret or output.

Adversarial models MPC schemes are designed to prevent successful attacks by an adversary $\mathcal{A}$ that tries to corrupt one or more parties which can then: (1) passively learn the secret values or information about the secret values,the protocol (e.g. randomness in the protocol) and the intermediate inputs/outputs; or (2) cause incorrect computations by actively involving in changing or communicating wrong input or intermediate values. One can classify security models based on the number of corrupted parties and based on their behavior.

Let us assume there are n parties $P = P_1, P_2 \dots P_n$ in total and $\mathcal{A}$ can corrupt any subset of parties of size t. If the number of corrupted parties is less than half the number of total

parties, i.e $t < n/2$, it is considered an *honest majority* setting as more than half the number of parties are honest (not corrupted). Whereas, in a *dishonest majority* setting at least half the number of parties are corrupted, i.e $t \geq n/2$.

A corrupted party can have two different behaviors, each of which may require a different type of security. A corrupted party who follows the protocol honestly but tries to learn information from the received messages or colludes with other corrupted parties to gather private information is a *semi-honest* or *passive* adversary. In the *semi-honest* security model, MPC protocols achieve security by preventing leakage of information among corrupted parties (which otherwise is leaked during collusion). They are quite efficient and best suited when the only concern is leakage of information. On the other hand, a *malicious* or *active* adversary, may arbitrarily deviate from the protocol. Such a corrupted party can control, modify or generate messages in addition to what a *passive* adversary can. The MPC protocols to detect attacks in the *malicious* security model are relatively expensive in terms of computations and communications as they preserve privacy and ensure correctness when a corrupted party can change the values. The protocols for malicious security ,in general, ensure that the output is guaranteed and correct if honest parties have received the output; and abort the protocols when cheating is detected in a dishonest majority setting. This ensures that privacy is always preserved.

3.2 Number representation

MPC schemes use various algebraic structures. In the MPC schemes that we use in our thesis, computations are performed over integers modulo M, i.e. $\mathbb{Z}_M$, where $M = 2 \ldots, M-1$ and $\mathbb{Z}_M$ is equipped with addition and multiplication modulo M . The choice of the computation domain represented by M affects the efficiency of the protocols used for the particular MPC scheme. When M is a prime number p, the computation domain is the field $\mathbb{Z}_p$ and when $M = 2^k$, the domain is the ring $\mathbb{Z}_{2^k}$.

Independent of the choice of computation domain, MPC protocols performs operations over discrete algebraic structures (finite fields and rings). So, it is important to map any

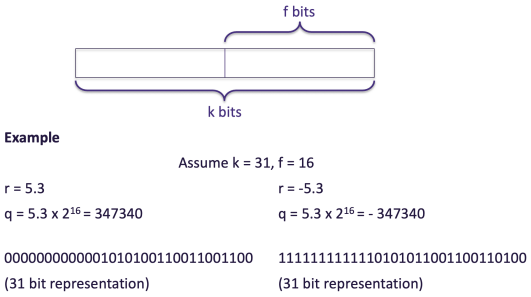

Figure 3.1: Fixed point representation: Mapping a real value to integer

input we want to perform computations on into such discrete structures. We consider these discrete structures as "integers" ($\mathbb{Z}_M$) for our work. In our application scenario however, *Alice* has her video preprocessed to represent it as a 4D-array of the real-valued pixels. Similarily *Bob* has a trained video classification model with the real-valued parameters. The real values should be converted to integers by *Alice* and *Bob* and then secret shared with the servers in S. These real values are generally represented in floating point notation on computers but performing MPC protocols on floating point notation is expensive.

A commonly used representation for real values in MPC is a fixed-point representation [16]. The real numbers are represented as $\lfloor r.2^f \rfloor$ where r is the real value to be mapped and f is the length of the fractional part, also called precision. In other words, the real value r is mapped to an integer q scaled by a factor of 2^{-f}. We can assume that the zero point is '0'.

$$r = 2^{-f}(q) \tag{3.1}$$

k is the total precision or the number of bits used to define q using twos-complement notation.

With $f = 16$, 5.3 will be, for example, converted to 347340 and then secret shared. Similarly, -5.3 will be converted as -347340 and then secret shared. As we explain in Section 3.3 secure addition, subtraction, and comparison of fixed-point numbers can be performed similar to as integers. Multiplication performed on two fixed point numbers generates extra f bits leading to a product upscaled by (2^{-2f}). This requires downscaling so that the product

can be represented as per the notation mentioned in the above equation. For our work, we employ the deterministic protocol by Dalskov et al. [25] for computations over $\mathbb{Z}_{2^k}$

3.3 MPC schemes and settings used

In our thesis, as shown in Figure 1.1, we consider three parties who engage in secure multi-party computations. We consider an *honest majority* setting where corrupted parties can be either *semi-honest* or *malicious*. Such a 3PC honest majority setting is growing popular in PPML literature [25, 47, 74, 82] because of availability of some efficient protocols for this setting. In other words, we consider a security model for $n = 3$ and $t = 1$ ($t < n/2$) and use $(1, 3)$ threshold secret sharing schemes and MPC protocols that work over a ring ($\mathbb{Z}_{2^k}$). This security model thus can tolerate one corrupted party and needs at the least 2 parties to reconstruct the secret. The protocols that we develop for our thesis in Chapter 4 can also be used in other settings provided the underlying building blocks and schemes are chosen accordingly.

Replicated secret sharing An efficient kind of MPC scheme for 3 party computation (3PC)is replicated secret sharing. In this kind of MPC scheme, a value x in $\mathbb{Z}_M$ is secret shared among three servers (parties) S_1, S_2, and S_3, by generating uniformly random shares such that

$$x = x_1 + x_2 + x_3 \mod M \tag{3.2}$$

where $x_i \in \mathbb{Z}_M$ are the secret shares of x. To secret share a value x, the owner of the secret x uniformly selects two random values x_1, x_2 in the computation domain M. The third value is then calculated as $x_3 = x - x_1 - x_2 \mod M$.

Generally in a replicated secret sharing scheme, the secret shares are distributed as pairs such that S_i holds the ith pair. To elaborate, S_i, who is the dealer, sends x_j, x_{j+1} to $S_{j\neq i}$. (x_4 is regarded as x_1, thus all i, j are computed over modulo 3) and $S_{j\neq i}$ takes the two received shares as its own share. To reveal the secret share to all the servers, every S_i sends x_i to S_{i+1}, where all the parties can reconstruct the shares locally to obtain the secret. In case the secret is to be revealed to only a single party S_i (such as in Step 4 of Figure 1.1),

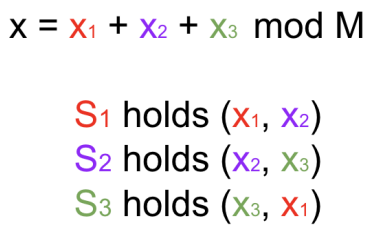

Figure 3.2: Replicated secret sharing among 3 parties

party S_{i-1} will send the share x_{i-1} to S_i who will reconstruct the secret locally. There are many variations in the notations and the scheme, but the basic idea behind this is that every party holds two shares of the secret and any two parties can come together to reconstruct the secret.

In our use-case, the owner of x (*Alice* or *Bob* initially and then on any of S_i) privately sends x_1, x_2 to S_1, x_1, x_3 to S_3 and x_2, x_3 to S_2 as shown in Figure 3.2. Every party (server) thus holds a replica of the secret share. Such distribution ensures that no server can alone infer or deduce any information about x. The secret can be reconstructed by combining secret shares from at least any two servers. Once the initial secret sharing is completed by *Alice* or *Bob*, the servers continue to perform computations on the secret shared data. Linear operations such as addition, addition with a constant, and multiplication with a constant can be performed locally by all the servers.

Addition:

Let us assume that $z = x + y$ is to be computed where x and y are secrets. If x is split as $x = x_1 + x_2 + x_3 \mod M$ and y is split as $y = y_1 + y_2 + y_3 \mod M$, then as per the replicated secret sharing scheme, S_1 will hold x_1, x_2, y_1, y_2, S_2 will hold x_2, x_3, y_2, y_3 and S_3 will hold x_3, x_1, y_3, y_1. They will perform the addition on the secret shares locally, and as a result will hold the secret shares of $z = z_1 + z_2 + z_3 \mod M$. In other words, all parties compute $[\![x]\!] + [\![y]\!]$ generating $[\![z]\!]$ i.e secret share pairs from the set $(x_1 + y_1, x_2 + y_2, x_3 + y_3)$. This is illustrated in Figure 3.3 where at the end of this protocol, the servers will hold $[\![z]\!]$

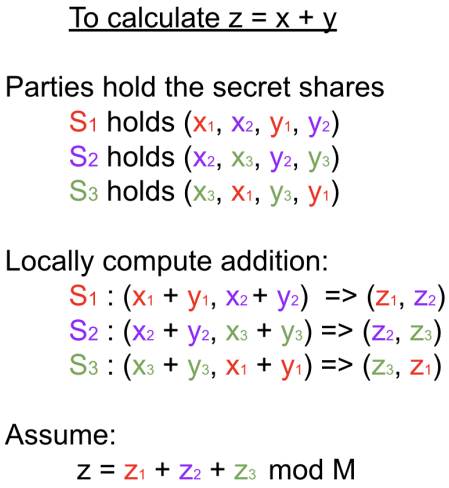

Figure 3.3: Addition in replicated secret sharing scheme

as if z was directly secret shared to the servers.

Multiplying with a public constant:

To multiply by a public constant c, all the parties can locally multiply their secret shares with c as illustrated in Figure 3.4. The parties hold secret shares of the product.

Adding a public constant:

Adding or subtracting a public constant value can be done in a similar way as illustrated in Figure 3.5, but the secret share pairs will be from the set $(x_1 \pm c, x_2, x_3)$.

Multiplication:

To compute the product of two secret shared values $z = x \cdot y$ (in the equation below) where x and y are secrets, the parties need to communicate. As before, x is split as $x = x_1 + x_2 + x_3 \mod M$ and y is split as $y = y_1 + y_2 + y_3 \mod M$, and S_1 will hold x_1, x_2, y_1, y_2, S_2 will hold x_2, x_3, y_2, y_3, and S_3 will hold x_3, x_1, y_3, y_1.

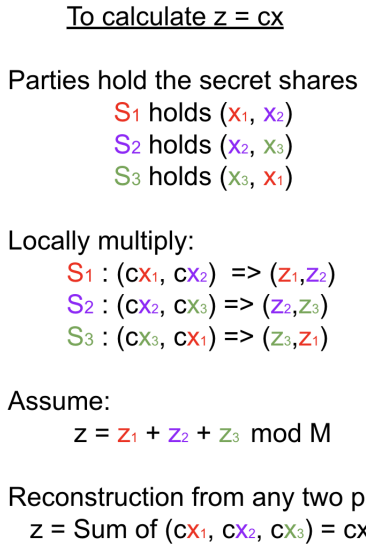

Figure 3.4: Multiplication with a public constant in replicated secret sharing scheme

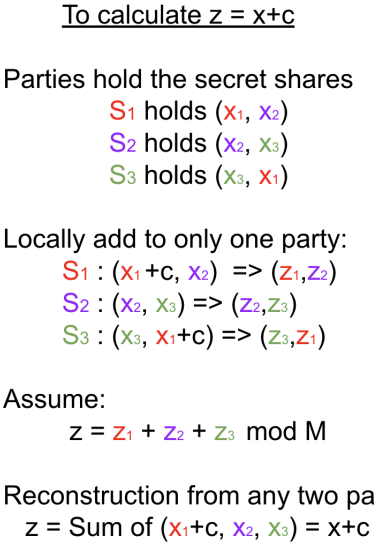

Figure 3.5: Addition of a public constant in replicated secret sharing scheme

$$
\begin{aligned}
z &= (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) && (3.3)\\
&= x_1y_1 + x_1y_2 + x_1y_3 && (3.4)\\
&\quad + x_2y_1 + x_2y_2 + x_2y_3 && (3.5)\\
&\quad + x_3y_1 + x_3y_2 + x_3y_3 && (3.6)
\end{aligned}
$$

The protocol proceeds as per the following steps:

- S_i calculates $v_i = x_iy_i + x_iy_{i+1} + x_{i+1}y_i$ locally.[1]
- A triple of random values (u_1, u_2, u_3) over computational domain $\mathbb{Z}_M$ are selected such that $u_1 + u_2 + u_3 = 0$. Each party knows only one of these values, i.e. S_1 will have u_1 only. These can be computed locally by the individual servers by generating the random shares of '0' using pseudo random functions, and secret sharing them using a simple additive secret sharing scheme. Having such values helps in randomizing the shares of v.
- S_i locally adds u_i to v_i to generate z_i.
- S_i sends z_i to S_{i-1}. This is done because after the above step each party holds one new share, but in replicated secret sharing we need that secret shares are held as pairs, i.e all parties must hold two out of the three generated shares.
- Now S_i holds z_i and z_{i-1} similar to how it held the secret shares of x.

This is illustrated in Figure 3.6 and compiled in the equations below.

[1]Consider x_4 as x_1 and y_4 as y_1 and S_0 as S_3

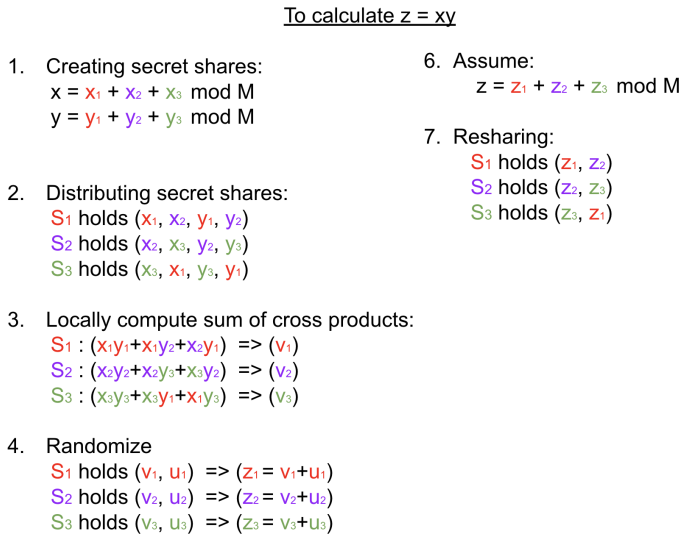

Figure 3.6: Multiplication in replicated secret sharing scheme.

$$
\begin{aligned}
z &= (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) \\
&= x_1y_1 + x_1y_2 + x_1y_3 + x_2y_1 + x_2y_2 + x_2y_3 + x_3y_1 + x_3y_2 + x_3y_3 \\
&= (x_1y_1 + x_1y_2 + x_2y_1) + (x_2y_2 + x_2y_3 + x_3y_2) + (x_3y_3 + x_3y_1 + x_1y_3) \\
&= v_1 + v_2 + v_3 \\
&= v_1 + v_2 + v_3 + 0 \\
&= v_1 + v_2 + v_3 + (u_1 + u_2 + u_3) \\
&= (v_1 + u_1) + (v_2 + u_2) + (v_3 + u_3) \\
&= z_1 + z_2 + z_3
\end{aligned}
$$

Passive security In this security setting, we adopt the efficient MPC schemes by Araki et al. [4]. As seen above, it requires a total communication for sending only three elements (z_i) among the parties for the multiplication protocol (Step 7 Resharing in Figure 3.6 where each party sends only one element). We follow the name convention as used in [25] to denote the MPC schemes for replicated sharing in the 3PC, honest majority setting, namely

*Replicated*2k for a ring domain

Malicious security All descriptions given above about replicated sharing are for the passive security setting. Unlike in the passive security setting, in the active security setting a corrupted party may deviate from the protocol instructions. MPC schemes based on replicated sharing exist for the active security setting as well. They can be thought of as extensions of the corresponding MPC schemes for the passive security setting. Privacy, i.e. no leakage of private information can be guaranteed by the schemes used for passive security. Additionally, to detect when a party is deviating from the protocol, for the active security setting, each secret share is tagged with another shared value known as MAC (messsage authentication code). MACs, in other words, authenticate each secret share to ensure that the parties are sending the correct values.

For the active security setting in the ring domain, we use the approach by Eerikson et al. [28]. Following the convention introduced in [25], we call the corresponding MPC scheme for the 3PC, honest majority setting with malicious adversaries *PsReplicated2k*. This approach employs 'abort-on-detect' and does post-processing by adapting techniques from [51] to ring domains.

The above protocols, independent of the security setting, are usually divided in an online and an offline phase. The offline phase involves any preprocessing steps (such as generating random values) or computations that can be done independent of the inputs (x and y are inputs in $z = x + y$), whereas computations in the online phase completely depend on the inputs and cannot be executed prior to receiving inputs unlike the offline phase. We note that security of the protocols that we use comes from the SPDZ's construction.

MPC primitives Above we have briefly described how secret sharing based MPC schemes can be used to perform basic operations such as addition and multiplication. We use MPC protocols available in the literature, that build upon these basic cryptographic primitives, for the following operations required for our work:

- Secure matrix multiplication π_{DMM} [25]: This protocol takes secret sharings $[\![A]\!]$ and $[\![B]\!]$ (parties hold these secret shares) of matrices A and B as input and performs $C = A \times B$.

The parties after this protocol hold the secret shares $[\![C]\!]$. π_{DMM} extends the basic secure multiplication protocol explained earlier in this section, which we denote as π_{DM} henceforth.

- Secure comparison protocol π_{LT} [14]: This protocol takes secret sharings $[\![x]\!]$ and $[\![y]\!]$ (parties hold these secret shares) of integers x and y. The result is secret sharing of 1 if $x < y$, and a secret sharing of 0 otherwise.

- Secure argmax π_{ARGMAX}: This protocol accepts secret shares of a vector of integers and returns a secret sharing of the index at which the vector has the maximum value. π_{ARGMAX} is constructed using π_{LT}.

- Secure RELU π_{RELU} [25]: The protocol starts with the parties holding secret shares of z. At the end of the protocol, the parties have a secret sharing of the value $\max(0, z)$. π_{RELU} is constructed from π_{LT}, followed by a secure multiplication to either keep the original value z or replace it by zero in an oblivious way.

- Secure division π_{DIV}: We directly adopt a well-known secure division protocol available in the literature [15].

3.4 Video classification

Videos can be thought of as sequential data as they can be considered as a stack of images called frames, where subsequent frames in the video are correlated with respect to their semantic contents and temporal characteristics. Deep learning using artificial neural networks (ANNs) [89] is widely used in performing video classification. It is possible to use ANN models such as RNNs (recurrent neural networks), especially LSTMs (long short term memory), and CNNs (Convolutional neural networks, also called ConvNets) [43] by fusing temporal information from consecutive frames, using 2D convolutions as used for images or taking a rolling average of classifications. This is referred to as "utilizing the spatial context". Another pioneering approach for video classification is using two separate networks – one for

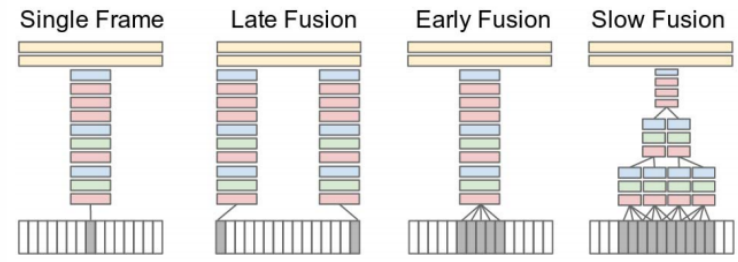

Figure 3.7: Single Stream Approach [43]

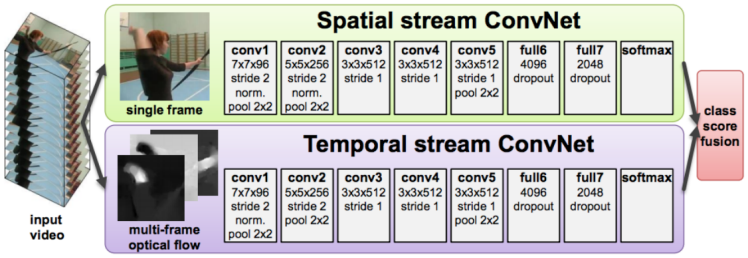

Figure 3.8: Two Stream Approach [79]

spatial context and one for motion context [79]. The above two popular approaches can be described as below

- Single stream network [43] : This network tries to fuse temporal information from consecutive frames into pre-trained 2D convolution networks. Various fusion methods are shown in Figure 3.7. In this thesis we adopt the *single frame based approach*, which we describe in more detail below.
- Two stream network: The above approach could not capture motion features. Simonyan et al. proposed to use two different networks [79], as illustrated in Figure 3.8. The motion features are tracked using optical flow information. The spatial network is a pre-trained convolution network which takes a single frame as input, while the temporal stream network which uses bi-directional optical flow. These two are combined with a classifier such as a support vector machine (SVM).

Many ANN architectures such as C3D, Conv3D, and LRCN have evolved for video classification built over these approaches. Popular techniques used for video classification in the literature are:

- Classifying one frame at a time with a Convolution Neural Network (ConvNet) [43]
- Using a time-distributed ConvNet and passing the features to an RNN, in one network [93, 88]
- Using a 3D convolutional network [81]
- Extracting features from each frame with a ConvNet and passing the sequence to a separate RNN [93, 88]
- Extracting features from each frame with a ConvNet and passing the sequence to a separate Multi-layer Perceptron (MLP)

To securely classify a video using MPC, we need to choose a "MPC-friendly" architecture. By "MPC-friendly", we mean an architecture which, when trained into a model, can infer a video using operations that can be expressed can be performed using existing MPC protocols available in the literature or can be constructed as an MPC protocol. To have such a model for our work, we propose to use the *single frame based approach* with frames selected in regular intervals over the video. Each frame is proposed to be individually classified using a CNN. The class labels inferred for the individual frames will then be aggregated over the frames based on the confidence in the inferred labels.

Proposed video classification pipeline We propose the following pipeline for classifying a video illustrated in Figure 3.9 for a video with 100 frames and 7 classes where each frame is selected at an interval of 15.

1. Selecting a subset of frames from the video: A video has spatial as well as temporal information. The individual frames in the video can be used to extract the spatial information. The number of frames in a video is often high, while the main content is concentrated in a subset of the frames. Efficient and accurate video classifiers typically select a small number of frames from the video and ignore the rest. In case of emotion recognition from

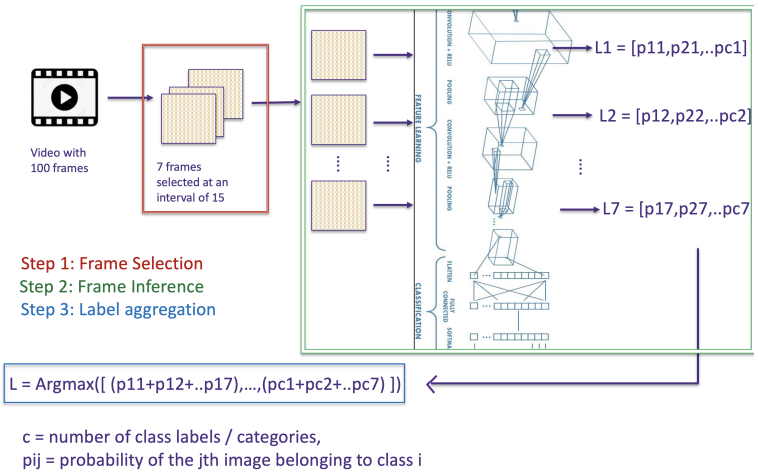

Figure 3.9: Proposed video classification pipeline

video, such selected sets of frames are known as apex frames which contain the most expressive emotional state. For our work, we sample frames at predefined equal intervals for classification.

2. Classifying each frame: We use a CNN model to classify each frame by treating the frame as a single image that needs to be classified. We use the spatial information in each frame and get class labels for individual frames.
3. Label aggregation: There are various techniques used to aggregate labels from a collection of individual labels. We use a technique that outputs a class label which is the dominant class label across the video obtained by averaging the probability of all the inferred class labels for the individual frames.

3.4.1 Convolutional neural networks

Convolutional Neural Networks (CNNs or ConvNets) are deep learning models which take an ordered input such as an image and assign learned importance (weights and biases) to the various aspects/objects/features of the input as shown in Figure 3.11. They can further classify new inputs based on the learned model parameters as shown in Figure 3.10. The

preprocessing or feature extraction required for CNNs is much lower compared to more traditional featureful approaches such as a decision tree model or SVMs (Support Vector Machines). CNNs are layered architectures that learn different features in each layer and are the state-of-art to classify images and thus can be used to classify individual frames in a video.

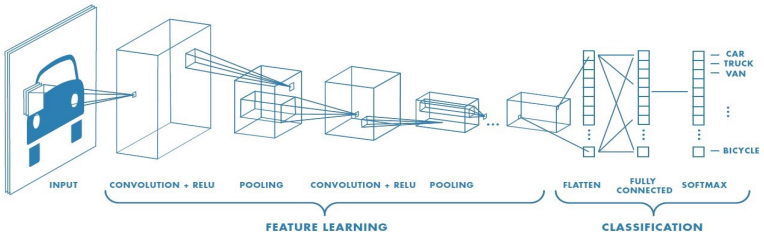

Figure 3.10: Inference with CNN[2]

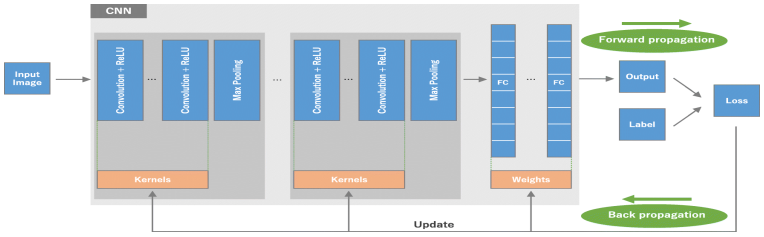

Figure 3.11: Training of CNN [91]

For our work, we propose to use a standard ConvNet. Such an architecture has one or more feature extraction blocks, each of which consists of a convolutional layer followed by an activation layer such as RELU and an optional pooling layer. These feature extraction layers are then followed by classification layers which consist of fully connected (FC) layers which are then followed by an activation function such as RELU except for the last layer. The last layers uses an activation function based on the type of classification problem. For our specific problem of multi-class classification, we use Softmax as the activation function in the last fully connected layer.

The major operations in a CNN are categorized as linear and non-linear, and the inference

pipeline has to perform these operations. For secure inference, we need to implement and optimize MPC protocols to perform the operations described below.

1. **Convolution layers**

   Convolution can be described as a process where we take a small matrix of numbers (called filter/kernel) and pass it over the input image and transform it based on the values from filter. This transformed output is called the feature map. In this layer, we will have an input image and several filters. The values of the filters/kernels are learnt during the training. After placing our filter over a selected pixel of the input image, we take each value from kernel and multiply them in pairs with corresponding values from the image. Finally we sum up everything and put the result in the right place in the output feature map. This is shown in Figure 3.13. In short, we can say that this layer generates convolutional product between the image and the filter as a 2D matrix where each element is the sum of the element wise multiplication of the filter and the selected subcube of the input.

   If the single input at this layer has dimensions $h \times w \times c$, where h is the height of the frame, w is the width of the frame and c is the number of channels used to represent the frame, and the dimensions of a filter are $h_f \times w_f \times c$ where h_f and w_f are the height and width of the filter with stride s and with padding p, then the dimensions of the output are $(h - h_f + 2p)/s + 1 \times (w - w_f + 2p)/s + 1 \times F_f$ where F_f is number of filters in this layer. Assuming biases as b for each filter, the total number of parameters are $(h_f \times w_f \times c + 1) \times F_f$. Figure 3.13 shows how a convolution operation works for a 6x6x3 input with a 3x3x3 filter and produces a 4x4x1 feature map assuming s=1 and p=0.

   The convolution operation is a linear operation and can be represented as

$$conv(I, K)_{x,y} = \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{l=1}^{c} K_{i,j,l} I_{x+i-1,y+j-1,l} + b_K \tag{3.7}$$

   where I is the input image, K is filter/kernel, b_K is the bias for the Kth filter.

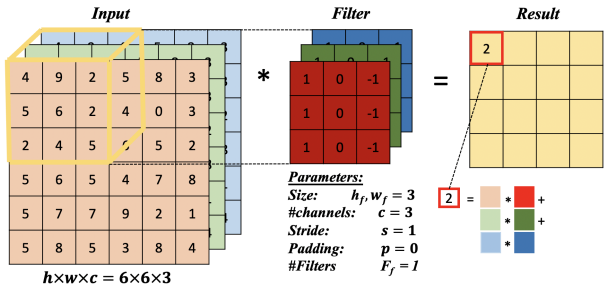

Figure 3.12: Example for convolution operation

2. **FC layers**

   A fully connected layer (FC) is a collection of finite number of neurons(perceptrons) which takes in input a vector and returns another one. If the input at this layer is a vector v_i with dimension n_i and the output vector is v_o with dimension n_o (n_o will be same as number of neuraons in this layer) and W is a weight matrix with dimensions $n_o \times n_i$ and a bias b with dimension n_o, then operations at this layer are linear and can be expressed as

$$v_o = W \cdot v_i + b \tag{3.8}$$

   The total number of parameters that are learned by the CNN are $(n_o \times n_i) + n_o$. The figure below illustrates this for an input vector of length 6 and an FC layer with 4 neurons generating an output of length 4.

3. **Pooling layers**

   Average Pooling finds the average value of the feature map of a given pool size. Average Pooling can be expressed as the addition of all the values in the given window divided by the pool size of the feature map. An input feature map of dimension $h \times w \times c$ with stride s and pool size $p \times p$ will generate an output of size $h/p \times w/p \times c$ by averaging the values in the feature map in each window of size $p \times p$ as illustrated in Figure 3.14.

4. **Activation functions**

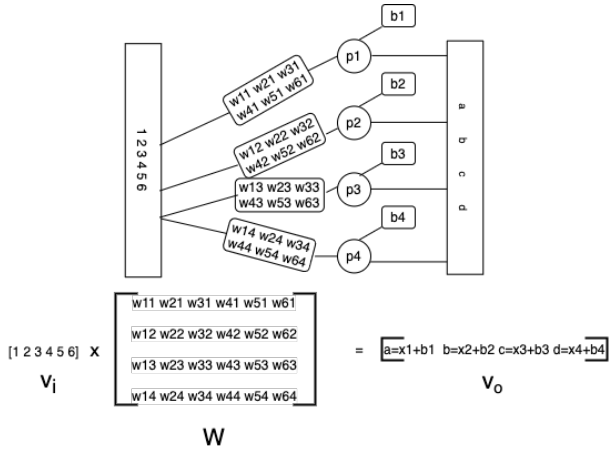

Figure 3.13: Example for fully connected layer operation

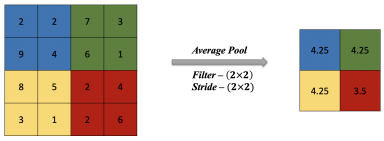

Figure 3.14: Example for average pooling

RELU is a non-linear function used in intermediate layers in general for CNN architectures for images and is defined as below for an input x.

$$f(x) = \begin{cases} x, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases} \tag{3.9}$$

Softmax, shown in Figure 3.16, is a non-linear function generally used as the activation function in the output layer for multi class classification problems . The Softmax function takes as input a vector $(u_1, u_2, \ldots, u_C)$ of length C and maps it to a vector $(\sigma(u_1), \sigma(u_2), \ldots, \sigma(u_C))$ with

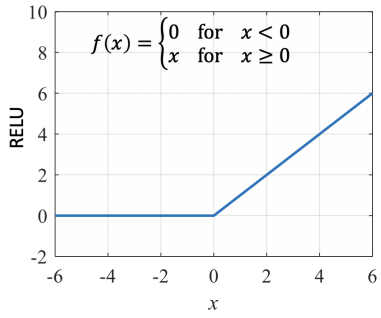

Figure 3.15: Activation functions: RELU

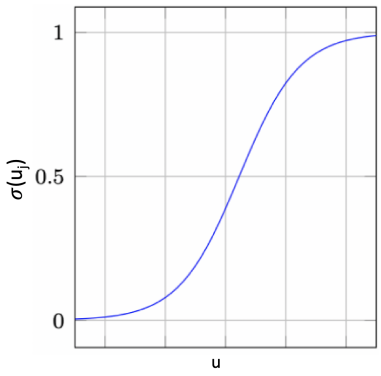

Figure 3.16: Activation functions: SoftMax

$$\sigma(u_i) = \frac{\exp(u_i)}{\sum_{j=1}^{C} \exp(u_j)} \tag{3.10}$$

for $i = 1, \ldots, C$. In a neural network trained to solve a multi-class classification problem with C classes where Softmax is used as the activation function on the output layer, $(u_1, u_2, \ldots, u_C)$ are referred to as the *logits* for each of the C class labels, and $(\sigma(u_1), \sigma(u_2), \ldots, \sigma(u_C))$ is interpreted as a probability distribution over the class labels. Indeed, the Softmax activation function normalizes the logits to values between between 0.0 and 1.0 that sum up to 1.0, thus providing a valid probability distribution of all the class labels.

Chapter 4

METHODOLOGY

We assume that *Alice*, who owns a video $\mathcal{V}$, requests *Bob* to classify the video, who owns a frame selection matrix $\mathcal{B}$ and a trained convolutional neural network model $\mathcal{M}$ for image classification. Neither of them are willing to disclose the contents of their data, except for the dimensions of the video and the architecture of the model (but not the trained model parameters). Both of them outsource the computations to a set S of three servers by secret sharing their data converted to integers modulo M. These servers or parties jointly perform computations to compute secret shares of the final class label for the video to be sent to *Alice*. To elaborate, the servers perform single-frame based video classification securely by

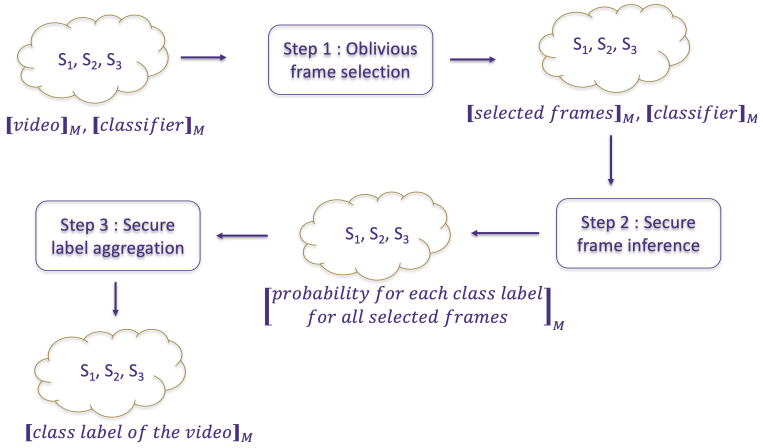

Figure 4.1: Proposed secure video classification pipeline

(1) selecting required frames from the video in an oblivious manner using $\mathcal{B}$; (2) securely classifying the selected frames using $\mathcal{M}$; and (3) securely aggregating the individual labels for the selected frames to obtain a single class label for the video $\mathcal{V}$. The process is illustrated

in Figure 4.1.

4.1 Oblivious frame selection

At this point, *Alice* has preprocessed her video to represent it as a 4D-tensor A of real numbers represented using fixed point notation and secret shared it with the servers. A is of shape $N \times h \times w \times c$, where N is the total number of frames in the video, h is the height of the frame, w is the width of the frame and c is the number of channels used to represent the frame. The dimensions N, h, w, and c are known to *Bob* and S but not the content of each frame.

The frame selection criterion that *Bob* uses while training the model in-the-clear is to select all frames in a video at a distance d apart, i.e. the frames with indices 0, d, $2d$, ... are selected. While securely classifying a new video, *Bob* uses the same criterion for selecting the frames. The parameter d is only known to *Bob*, who uses it to construct a vector b of length n which will contain the indices of the frames to be selected. In this way, *Alice* and S will have no knowledge of which frames *Bob* intends to select thus preventing *Alice* from inserting any malicious frames at these index positions. Also, once *Bob* has selected these frames, he and S have no knowledge of the content in these frames.

We make use of matrix multiplication to perform oblivious frame selection as illustrated in Figure 4.2. For this purpose, *Bob* will convert the vector b into a one-hot-encoded matrix $\mathcal{B}$ of size $n \times N$ and secret shares it with the servers (parties) in S. The parties now engage in joint computations as per Protocol 1 to get resultant frames in a 4D-tensor F. To begin with, A is flattened to a size of $N \times (h \times w \times c)$ as A_{flat}. Secure matrix multiplication of $\mathcal{B}$ $\times$ A_{flat} results in a matrix F_{flat} of size $n \times (h \times w \times c)$, which is then expanded to form F of size $n \times h \times w \times c$.

4.2 Secure frame classification

The parties after execution of π_{FSELECT} now hold the secret shares of n selected frames which are further to be securely classified using a ConvNet model. *Bob* secret shares the real-

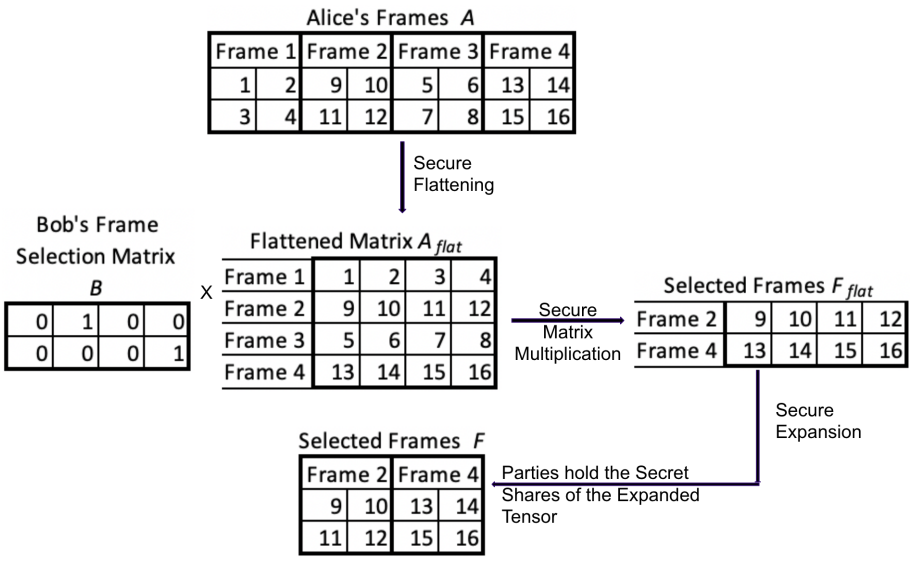

Figure 4.2: Illustration of oblivious frame selection. The assumption is made that *Alice* has 4 frames in total, each of size $2 \times 2 \times 1$, and *Bob* needs to select 2 frames, namely Frames 2 and 4. *Alice* has a tensor A of size $4 \times 2 \times 2 \times 1$ and *Bob* has a 2D-matrix $\mathcal{B}$ of size 2×4. A is flattened securely to form A_{flat} of size 4×4. A secure matrix multiplication $\mathcal{B} \times A_{\mathsf{flat}}$ is performed resulting in F_{flat}, a 2×4 matrix holding the 2 selected frames. This matrix is then expanded to matrix F of size $2 \times 2 \times 2 \times 1$.

Protocol 1: Protocol π_{FSELECT} for oblivious frame selection

Input : A secret shared 4D-array $[\![A]\!]$ of size $N \times h \times w \times c$ with the frames of a video; a secret shared frame selection matrix $[\![\mathcal{B}]\!]$ of size $n \times N$. The values N, h, w, c, n are known to all parties.

Output: A secret shared 4D-array F of size $n \times h \times w \times c$ holding the selected frames

1 $[\![A_{\mathsf{flat}}]\!] \leftarrow \mathsf{RESHAPE}([\![A]\!], N \times h \times w \times c, N \times (h \times w \times c))$

2 $[\![F_{\mathsf{flat}}]\!] \leftarrow \pi_{\mathsf{DMM}}([\![\mathcal{B}]\!], [\![A_{\mathsf{flat}}]\!])$

3 $[\![F]\!] \leftarrow \mathsf{RESHAPE}([\![F_{\mathsf{flat}}]\!], n \times (h \times w \times c), n \times h \times w \times c)$

4 **return** $[\![F]\!]$

valued parameters of $\mathcal{M}$, a 2D-ConvNet model, in fixed point notation to S. $\mathcal{M}$ has an "MPC-friendly" architecture (cfr. Section 3.4.1) in which all operations except for the last activation layer can be represented as a function of basic "MPC operations/primitives" that are well-known in the literature as described in Section 3.3. More in detail, convolution and fully connected layers can be represented as a function of additions and multiplications. The average pooling layer can be expressed using additions and π_{DIV}. The RELU activation function can be expressed using π_{RELU}. These operations for ConvNet can be performed using efficient MPC protocols available in the literature [25]. We refer to the sequence of these protocols as defined by the architecture of $\mathcal{M}$ leaving out the activation function in the last layer as π_{FLOGIT}. Thus after execution of π_{FLOGIT}, the parties in S hold the secret shares of the 'logits' as computed using $\mathcal{M}$ for the individual frame.

Softmax is generally used as the activation function in the last layer for multi-label classification problems. It is a non-linear, monotonic and differential function and is thus useful for training purposes with gradient descent based optimizations. The output of the Softmax operation gives a valid probability distribution of all the class labels where the 'logits' are normalized, making it an apt choice for multi-class classification problems. Though Softmax

function is important during training, it is a costly function to implement using MPC protocols as it involves expensive operations such as division and exponentiation (See Equation 4.1). Given the logits $(u_1, u_2, \ldots, u_C)$ for each of the C class labels as input, Softmax converts them into the output $(\sigma(u_1), \sigma(u_2), \ldots, \sigma(u_C))$ with

$$\sigma(u_i) = \frac{\exp(u_i)}{\sum_{j=1}^{C} \exp(u_j)} \tag{4.1}$$

for $i = 1, \ldots, C$.

A few works on privacy-preserving inference propose to perform the Softmax operation in an unencrypted manner by revealing the logits [53], while other works which require only the class label replace Softmax with Argmax [25, 8]. For our task of video classification, we however require the inferred probabilities of all classes to aggregate the labels to derive the final class label for the video (cfr. 3.4).

For this purpose, we propose to replace the Softmax function by an approximate function (using Equation 4.2) provided by Mohassel and Zhang [66]

$$f(u_i) = \begin{cases} \dfrac{\mathrm{RELU}(u_i)}{\sum_{j=1}^{C} \mathrm{RELU}(u_j)}, & \text{if } \sum_{j=1}^{C} \mathrm{RELU}(u_j) > 0 \\ 1/C, & \text{otherwise} \end{cases} \tag{4.2}$$

for $i = 1, \ldots, C$, where $(u_1, u_2, \ldots, u_C)$ denote the logits for each of the C class labels, and $(f(u_1), f(u_2), \ldots, f(u_C))$ is the computed probability distribution over the class labels.

The MPC protocol π_{SOFT} for the above approximate function is presented in Protocol 2, which takes securely computed secret shares of the logits for a single frame and calculates a secret shared probability distribution for the class labels ensuring that no information is leaked, not even based on the number and type of operations performed such as in Line 7 and 10.

We define π_{FINFER} as the MPC protocol to securely infer a frame using π_{FLOGIT}, which

Protocol 2: Protocol π_{SOFT} for computing the approximate Softmax

Input : A secret shared list $[\![logits]\!]$ of logits of size C, where C is total number of class labels and known to all parties

Output: A secret shared list $[\![SM_{\mathsf{approx}}]\!]$ of size C of probabilities for the class labels

$[\![X_{\mathsf{relu}}]\!] \leftarrow \pi_{\mathsf{RELU}}\ ([\![logits]\!])$
$[\![Sum_{\mathsf{relu}}]\!] \leftarrow 0$
for $j \leftarrow 1$ **to** C **do**
    $[\![Sum_{\mathsf{relu}}]\!] \leftarrow [\![Sum_{\mathsf{relu}}]\!] + [\![X_{\mathsf{relu}}[i]]\!]$
end
$[\![cn]\!] \leftarrow \pi_{\mathsf{LT}}\ (0, [\![Sum_{\mathsf{relu}}]\!])$
$[\![denom]\!] \leftarrow \pi_{\mathsf{DM}}\ ([\![cn]\!], [\![Sum_{\mathsf{relu}}]\!]) + \pi_{\mathsf{DM}}\ ((1 - [\![cn]\!]), C)$
$[\![denom\_inv]\!] \leftarrow \pi_{\mathsf{DIV}}\ (1, [\![denom]\!])$
for $i \leftarrow 1$ **to** C **do**
    $[\![numer]\!] \leftarrow \pi_{\mathsf{DM}}\ ([\![cn]\!], [\![X_{\mathsf{relu}}[i]]\!]) + (1 - [\![cn]\!])$
    $[\![SM_{\mathsf{approx}}[i]]\!] \leftarrow \pi_{\mathsf{DM}}\ ([\![numer]\!], [\![denom\_inv]\!])$
end
return $[\![SM_{\mathsf{approx}}]\!]$

securely gets the logits for a single frame, and then using π_{SOFT} to securely get the probability distribution over the class labels for the frame f.

4.3 Secure label aggregation

As illustrated in Figure 4.3, we classify a video by selecting the class label which has the highest probability sum. The idea behind this is to consider the dominant class labels throughout the video. To elaborate further, we classify a video by first calculating the sum of probabilities for each class across all the frames, then taking the argmax of the sums, thus returning the class label for the video.

SM_{approx} for Frames

Labels →	1	2	3	4	5	6	7
Frame 1	0	0	0	0	0.28	0	0.72
Frame 2	0	0	0	0	0.55	0.45	0
Frame 3	0	0	0	0	0.83	0.17	0
Frame 4	0	0.21	0	0	0.48	0.31	0
$prob_{sum}$	0	0.21	0	0	2.14	0.93	0.72

Output Label L is 5

Figure 4.3: Illustration of label aggregation. Let us assume that $n = 4$ frames were selected for secure inference, and that there are $C = 7$ classes. SM_{approx} holds the inferred probability distribution over the class labels for each frame. Class label 5 is selected as the final label because it has the highest sum of probabilities across all classified frames.

We implement this securely as per Protocol $\pi_{\mathsf{LABELVIDEO}}$, which is described in Protocol 3. The steps in this protocol to classify a video $\mathcal{V}$ are:

1. Using π_{FSELECT} select the required frames;
2. For each frame get the probability distribution $[\![SM_{\mathsf{approx}}]\!]$ using π_{FINFER};
3. Sum up all these distributions index-wise into $[\![prob_{\mathsf{sum}}]\!]$;
4. Take the argmax of $[\![prob_{\mathsf{sum}}]\!]$ to get the secret shares of the final label L.

The parties in S now hold secrets shares $[\![L]\!]$, which are then sent to *Alice*, where the secret shares are reconstructed to reveal the final class label L for the video.

Protocol 3: Protocol $\pi_{\mathsf{LABELVIDEO}}$ for classifying a video securely based on the single-frame method

Input : A video $\mathcal{V}$ secret shared as a 4D-array $[\![A]\!]$, a frame selection matrix secret shared as $[\![B]\!]$, the parameters of the ConvNet model $\mathcal{M}$ secret shared as $[\![M]\!]$

Output: A secret share $[\![L]\!]$ of the video label

Let $[\![prob_{\mathsf{sum}}]\!]$ be a list of length C that is initialized with zeros in all indices.
$[\![F]\!] \leftarrow \pi_{\mathsf{FSELECT}}\ ([\![A]\!], [\![B]\!])$
for each $[\![F[j]]\!]$ **do**
    $[\![SM_{\mathsf{approx}}]\!] \leftarrow \pi_{\mathsf{FINFER}}\ ([\![M]\!], [\![F[j]]\!])$
    for $i \leftarrow 1$ **to** C **do**
        $[\![prob_{\mathsf{sum}}[i]]\!] \leftarrow [\![prob_{\mathsf{sum}}[i]]\!] + [\![SM_{\mathsf{approx}}[i]]\!]$
    end
$[\![L]\!] \leftarrow \pi_{\mathsf{ARGMAX}}\ ([\![prob_{\mathsf{sum}}]\!])$
return $[\![L]\!]$

Chapter 5

RESULTS

5.1 Dataset

We demonstrate the feasibility of our privacy-preserving approach to single-frame method based video classification for the task of emotion detection from videos using the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [57]. We use 1248 videos from this dataset corresponding to 7 different emotions: neutral (96), happy (192), sad (192), angry (192), fearful (192), disgust (192) and surprised (192). We use the video-only files from the 'speech' subset of RAVDESS dataset. The dataset consists of recordings of 24 actors, balanced in gender. Each actor expresses each emotion, except neutral, with two different intensities, namely strong and normal. Each emotion of an intensity is expressed by reading two different statements and each statement is read twice. This results in 52 video files per actor (= 6x2x2x2 + 1x1x2x2) giving a total of 1,248 videos for 24 actors. The videos in the RAVDESS dataset have on average a duration of 3-5 seconds with an aspect ratio of 16:9 and 30 frames per second. We observe that videos in the RAVDESS dataset have a total number of frames in the range of 90-150.

As mentioned in Section 5.2, we use the FER2013 dataset for pre-training which has only the above 7 emotions. Due to this, we leave out the 'calm' emotion from the original RAVDESS dataset as in [10].

The dataset is split into train and test datasets. The test dataset is formed by selecting all video files of actors 8,15 (selected randomly – one female and one male to have balanced gender) and another randomly selected 28 videos. The remaining videos are kept as the training dataset. This results in a training dataset of 1,116 videos and a test dataset of 132 videos.

5.2 Video preprocessing and model training

Preprocessing a video: The videos are preprocessed in-the-clear to form a 4D tensor of size $N \times h \times w \times c$ as mentioned in Section 4.1. OpenCV [9] is used to read the video into frames. Every frame is then preprocessed by detecting faces with a confidence greater than 98% using MTCNN [94], aligning, cropping and then converting it to gray-scale. Each processed frame is then resized to a size of 48×48, reshaped to a 4D-array and normalized by dividing each pixel value by 255.0. The same preprocessing technique is applied uniformly on the train and test dataset.

Training the model:[1] The architecture, shown in Figure 5.1 with about 1.48 million parameters, is pre-trained on the FER 2013 dataset [13][2] to extract facial features for emotion recognition. The feature-layers are trained with early stopping on the FER 2013 dataset with a batch size of 256 and using Adam Optimizer with the default parameter settings defined by Keras [20].

The model is then fine-tuned on the RAVDESS dataset. From the 1116 videos, 111 videos are selected randomly and kept aside as the validation set and the remaining videos are used for training. The frames are selected at a distance $d = 15$ i.e. every 15th frame in the video is selected for training. Training is done with early-stopping on a batch size of 64 using SGD optimizer with a learning rate 0.001, decay as 10^{-6}, and momentum as 0.9.

The trained model gives an accuracy of 56% on the test dataset. For secure inference, we replace the activation function in the last layer with an approximate function as mentioned in Section 4.2. This replacement results in an accuracy of 56.8% on test dataset. This is in-line with the results available in the literature (cfr. Section 2) for emotion recognition in videos on RAVDESS using spatial features or video-only modality.

---

[1] We used TensorFlow and Keras for training.

[2] https://datarepository.wolframcloud.com/resources/FER-2013

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 44, 44, 64)	1664
average_pooling2d (AveragePo	(None, 20, 20, 64)	0
conv2d_1 (Conv2D)	(None, 18, 18, 64)	36928
conv2d_2 (Conv2D)	(None, 16, 16, 64)	36928
average_pooling2d_1 (Average	(None, 7, 7, 64)	0
conv2d_3 (Conv2D)	(None, 5, 5, 128)	73856
conv2d_4 (Conv2D)	(None, 3, 3, 128)	147584
average_pooling2d_2 (Average	(None, 1, 1, 128)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 1024)	132096
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1024)	1049600
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 7)	7175

Total params: 1,485,831
Trainable params: 1,485,831
Non-trainable params: 0

Figure 5.1: Architecture of CNN in-the-clear

5.3 Experiments

Experimental Setup We consider S to consist of three servers for both active and passive corruption models for honest majority security settings, i.e. at most 1 server can get corrupted. We run our experiments for secure video classification on three co-located c5.4xlarge AWS EC2 instances with 16 vCPUs, 32.0 GiB Memory, connected over upto 10 Gbps link.

We implemented our proposed protocols from Chapter 4 in the MP-SPDZ framework [44]. We perform experiments over a ring $\mathbb{Z}_{2^k}$, with a value $k = 64$. We adopt replicated sharing scheme (Refer Chapter 3.3) for our implementation. We adopt protocols by Araki et al.[4] for passive corruption (*Replicated2k*) and protocols by Eerikson et al.[28] for active corruption (*PsReplicated2k*).

Results We report average metrics to classify a video securely in Table 5.1. This is calculated for a set of 10 videos with different numbers of frames N. We also report the average time taken for each step in the secure video classification pipeline and the average communication required per party. The accuracy of classifying a video securely is 56.8% on the test dataset.

In line with common knowledge, we observe that MPC protocols for the 3PC *passive* security setting are more efficient than the 3PC active security setting.

Overall in 3PC honest majority settings, it takes on average 13.85 seconds to classify a video in a semi-honest setting and 82.70 seconds in malicious setting. This shows that a faster responsive secure video classification system can be deployed depending on the trust assumptions of the application. The runtimes that we report here include time taken for preprocessing required for performing MPC protocols. We recall that the offline phase of computations is data-independent and involves all preprocessing steps. The computations required for offline phase can be done prior to the arrival of the video for classification resulting in faster response of a privacy-preserving video classification system than what is shown in Table 5.1. The MP-SPDZ framework, at the time of performing experiments, did not offer a mechanism to measure the time for the offline and the online phase separately, because both phases are intertwined in MP-SPDZ. It is an interesting direction for future

work to investigate the speed-up that can be achieved for video classification in real time through performing the offline phase fully in advance. We observe that most of the time required for classifying a video securely is spent on securely classifying an individual frame in the video, whereas securely aggregating the labels takes the least time. The frame selection step primarily uses only π_{DMM} and thus takes lesser time than frame inference. Inferring a single frame involves using a convolutional neural network with about 1.48 million parameters and is most computation intensive step in the complete pipeline for video classification. It involves expensive MPC operations such as π_{DMM}, π_{DM}, π_{RELU} and π_{DIV}. The final label aggregation majorly involves π_{ARGMAX} which is relatively cheaper. Inferring a single frame using a convolutional network architecture with 1.44 million parameters takes 0.7-8.9 seconds in our experimental setup using c5.4xlarge instance. Inferring an image of larger size (which is same as frame inference) with same MPC schemes and protocols with larger convolutional network architecture (with 4.24 million parameters) on more powerful machines (c5.9xlarge) takes 0.9-11.1 seconds [25]. The runtimes in Table 5.1 can be improved by using more powerful machines resulting in a more responsive privacy preserving video classification system. In a practical MLaaS scenario for secure video classification, which generally use powerful virtual machines, runtimes can be expected to be near to that of real time.

We also observe that malicious security settings require higher communication per party than passive security settings. This is because the protocols for malicious security requires additional communication head for MACs.

	Avg. Time VC (s)	Avg. Time FS (s)	Avg. Time single FI (s)	Avg. Time LA (s)	Avg. Comm. VC (MB)
Passive (3 PC)	13.85	0.37	0.69	0.0046	2556.2
Active (3 PC)	82.70	3.39	8.89	0.0047	19390.6

Table 5.1: **Metrics for classifying one video of duration 3-5 seconds** Average metrics are obtained over a set of 10 videos. VC - Time to classify one video ($\pi_{\mathsf{LABELVIDEO}}$), FS - Time for frame selection for one video (π_{FSELECT}), FI - Time to classify a single frame for one video averaged over selected frames (8) in the video (π_{FINFER}), LA - Time taken for label aggregation (π_{ARGMAX}) revealing class label of the video to Alice. Communication is measured per party

Chapter 6

CONCLUSION & FUTURE WORK

In this thesis, we have introduced, to the best of our knowledge, the very first MPC-based solution for classifying a video securely, i.e without leaking any private information. Our solution includes selecting frames from the video obliviously for further inference, securely inferring probability distributions over the class labels for these frames leveraging existing work, and aggregating these inferences securely to obtain the final class label of the video. We demonstrated the feasibility of our privacy-preserving approach in a use case for emotion recognition in videos. Our results show that we can classify a 3-5 second video securely in 13.8-82.7 seconds based on the protocols and security settings used. This runtime was obtained on modest hardware and includes both the offline (data independent) and online (data dependent) phases for the MPC protocols. In a deployment of our proposed MPC protocols for privacy-preserving video classification, the offline phase could be completed in advance, before the video that needs to be classified arrives. This would reduce the runtime and hence improve the responsiveness of the system substantially. Combined with the use of more powerful hardware, this would enable MPC based private video classification in (near) real time.

There are several interesting directions for further research. First of all, we can study runtimes for the online and offline phases separately to classify a video securely and improve on the video classification pipeline. Second, we can generalize the frame selection criteria using a machine learning algorithm rather than using indices predefined by *Bob*. This would enable to obliviously select frames based on the features present in each frame rather than just the indices and can thus give a generalized approach for frame selection. Such a model would automatically pick indices of the frames required and form the frame selection matrix.

Third, we can further explore to perform secure inference using more complex architectures and leverage transfer learning in the second step of our proposed approach.

BIBLIOGRAPHY

[1] M. Abdullah, M. Ahmad, and D. Han. Facial expression recognition in videos: An CNN-LSTM based model for video classification. In *International Conference on Electronics, Information, and Communication*, pages 1–3, 2020.

[2] N. Agrawal, A. Shahin Shamsabadi, M.J. Kusner, and A. Gascón. QUOTIENT: two-party secure neural network training and prediction, 2019.

[3] M.R. Ali, J. Hernandez, E.R. Dorsey, E. Hoque, and D. McDuff. Spatio-temporal attention and magnification for classification of Parkinson's disease from videos collected via the Internet. In *15th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 53–60, 2020.

[4] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 805–817, 2016.

[5] S. Avidan and M. Butman. Blind vision. In *European conference on computer vision*, pages 1–13. Springer, 2006.

[6] E. Bagheri, A. Bagheri, P.G. Esteban, and B. Vanderborgth. A novel model for emotion detection from facial muscles activity. In *Iberian Robotics conference*, pages 237–249. Springer, 2019.

[7] U. Bhattacharya, T. Mittal, R. Chandra, T. Randhavane, A. Bera, and D. Manocha. STEP: Spatial temporal graph convolutional networks for emotion perception from gaits. In *34th AAAI Conference on Artificial Intelligence*, pages 1342–1350, 2020.

[8] K. Bittner, M. De Cock, and R. Dowsley. Private speech characterization with secure multiparty computation. *arXiv preprint arXiv:2007.00253*, 2020.

[9] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[10] S. Bursic, G. Boccignone, A. Ferrara, A. D'Amelio, and R. Lanzarotti. Improving the accuracy of automatic facial expression recognition in speaking subjects with deep learning. *Applied Sciences*, 10(11):4002, 2020.

[11] M. Byali, H. Chaudhari, A. Patra, and A. Suresh. Flash: fast and robust framework for privacy-preserving machine learning. *Proceedings on Privacy Enhancing Technologies*, 2020(2):459–480, 2020.

[12] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[13] P.L. Carrier, A. Courville, I.J. Goodfellow, M. Mirza, and Y. Bengio. FER-2013 face database, Université de Montréal. `https://datarepository.wolframcloud.com/resources/FER-2013`, 2013.

[14] O. Catrina and S. De Hoogh. Improved primitives for secure multiparty integer computation. In *International Conference on Security and Cryptography for Networks*, pages 182–199. Springer, 2010.

[15] O. Catrina and S. De Hoogh. Secure multiparty linear programming using fixed-point arithmetic. In *European Symposium on Research in Computer Security*, pages 134–150. Springer, 2010.

[16] O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In *14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2010.

[17] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh. Astra: High throughput 3pc over rings with application to secure prediction.

[18] H. Chaudhari, R. Rachuri, and A. Suresh. Trident: Efficient 4pc framework for privacy preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS*, pages 23–26, 2020.

[19] F.Y. Chee. EU mulls five-year ban on facial recognition tech in public areas. *Reuters, Innovation and Technology*, Jan 16, 2020.

[20] F. Chollet et al. Keras. `https://keras.io`, 2015.

[21] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei. Faster cryptonets: Leveraging sparsity for real-world encrypted inference, 2018.

[22] R. Cramer, I. Damgård, and J.B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[23] R. Cramer, I.B. Damgard, and J.B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[24] A. Dalskov, D. Escudero, and M. Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. Cryptology ePrint Archive, Report 2020/1330, 2020.

[25] A. Dalskov, D. Escudero, and M. Keller. Secure evaluation of quantized neural networks. *Proceedings on Privacy Enhancing Technologies*, 2020(4):355–375, 2020.

[26] D. Deng, Z. Chen, Y. Zhou, and B. Shi. MIMAMO Net: Integrating micro-and macro-motion for video emotion recognition. *arXiv preprint arXiv:1911.09784*, 2019.

[27] M. D'Souza, J.F. Dorn, A. Dorier, C.P. Kamm, S. Steinheimer, F. Dahlke, C. EP Van Munster, B. MJ Uitdehaag, L. Kappos, and M. Johnson. Autoencoder as a new method for maintaining data privacy while analyzing videos of patients with motor dysfunction: Proof-of-concept study. *Journal of Medical Internet Research*, 22(5):e16669, 2020.

[28] H. Eerikson, M. Keller, C. Orlandi, P. Pullonen, J. Puura, and M. Simkin. Use your brain! Arithmetic 3PC for any modulus with active security. In *1st Conference on Information-Theoretic Cryptography (ITC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[29] D. Evans, V. Kolesnikov, and M. Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

[30] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[31] K. Hill. The secretive company that might end privacy as we know it. *The New York Times*, Jan 18, 2020.

[32] C. Hsu, Y. Zhuang, and C. Lee. Deep fake image detection based on pairwise learning. *Applied Sciences*, 10(1):370, 2020.

[33] M. Hu, H. Wang, X. Wang, J. Yang, and R. Wang. Video facial emotion recognition based on local enhanced motion history image and CNN-CTSLSTM networks. *Journal of Visual Communication and Image Representation*, 59:176–185, 2019.

[34] J. Imran, B. Raman, and A.S. Rajput. Robust, efficient and privacy-preserving violent activity recognition in videos. In *35th Annual ACM Symposium on Applied Computing*, page 2081–2088, 2020.

[35] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[36] M. Jaiswal and E. Provost. Privacy enhanced multimodal neural representations for emotion recognition. In *34th AAAI Conference on Artificial Intelligence*, pages 7985–7993, 2020.

[37] X. Jia, X. Zheng, W. Li, C. Zhang, and Z. Li. Facial emotion distribution learning by exploiting low-rank label correlations locally. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9841–9850, 2019.

[38] X. Jiang, M. Kim, K. Lauter, and Y. Song. Secure outsourced matrix computation and application to neural networks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1209–1222, 2018.

[39] W. Jiao, M.R. Lyu, and I. King. Real-time emotion recognition via attention gated hierarchical memory network. *arXiv preprint arXiv:1911.09075*, 2019.

[40] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasana. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium*, pages 1651–1669, 2018.

[41] L. Kalman. New European data privacy and cyber security laws – one year later. *Communications of the ACM*, 62:38–39, 2019.

[42] S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011.

[43] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[44] M. Keller. MP-SPDZ: A versatile framework for multi-party computation. Cryptology ePrint Archive, Report 2020/521, 2020. `https://eprint.iacr.org/2020/521`.

[45] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] N. Koti, M. Pancholi, A. Patra, and A. Suresh. Swift: Super-fast and robust privacy-preserving machine learning. *arXiv preprint arXiv:2005.10296*, 2020.

[47] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. CrypTFlow: Secure TensorFlow inference. In *41st IEEE Symposium on Security and Privacy*, 2020.

[48] D. Li, C. Rodriguez, X. Yu, and H. Li. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1459–1469, 2020.

[49] S. Li, K. Xue, C. Ding, X. Gao, D. Wei, T. Wan, and F. Wu. Falcon: A fourier transform based approach for fast and secure convolutional neural network predictions. *arXiv preprint arXiv:1811.08257*, 2018.

[50] S. Li, K. Xue, B. Zhu, C. Ding, X. Gao, D. Wei, and T. Wan. 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 (CVPR)*, June 2020.

[51] Y. Lindell and A. Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 259–276, 2017.

[52] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pages 36–54, 2000.

[53] J. Liu, M. Juuti, Y. Lu, and N. Asokan. Oblivious neural network predictions via MiniONN transformations. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631, 2017.

[54] J. Liu, Y. Xia, and Z. Tang. Privacy-preserving video fall detection using visual shielding information. *The Visual Computer*, pages 1–12, 2020.

[55] K. Liu, M. Zhu, H. Fu, H. Ma, and T. Chua. Enhancing anomaly detection in surveillance videos with transfer learning from action recognition. In *28th ACM International Conference on Multimedia*, page 4664–4668, 2020.

[56] X. Liu, W. Liu, M. Zhang, J. Chen, L. Gao, C. Yan, and T. Mei. Social relation recognition from videos via multi-scale spatial-temporal reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3574, 2019.

[57] S.R. Livingstone and F.A. Russo. 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.

[58] S. Manna, S. Ghildiyal, and K. Bhimani. Face recognition from video using deep learning. In *5th International Conference on Communication and Electronics Systems (ICCES)*, pages 1101–1106, 2020.

[59] E. Mansouri-Benssassi and J. Ye. Synch-graph: multisensory emotion recognition through neural synchrony via graph convolutional networks. In *34th AAAI Conference on Artificial Intelligence*, 2020.

[60] Ø. Meinich-Bache, S. L. Austnes, K. Engan, I. Austvoll, T. Eftestøl, H. Myklebust, S. Kusulla, H. Kidanto, and H. Ersdal. Activity recognition from newborn resuscitation videos. *IEEE Journal of Biomedical and Health Informatics*, 24(11):3258–3267, 2020.

[61] Z. Meng, S. Fu, J. Yan, H. Liang, A. Zhou, S. Zhu, H. Ma, J. Liu, and N. Yang. Gait recognition for co-existing multiple people using millimeter wave sensing, Apr. 2020.

[62] Z. Meng, S. Fu, J. Yan, H. Liang, A. Zhou, S. Zhu, H. Ma, J. Liu, and N. Yang. Gait recognition for co-existing multiple people using millimeter wave sensing. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 849–856, 2020.

[63] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R.A. Popa. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium*, 2020.

[64] T. Mittal, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha. M3ER: Multiplicative multimodal emotion recognition using facial, textual, and speech cues. In *34th AAAI Conference on Artificial Intelligence*, pages 1359–1367, 2020.

[65] T. Mittal, P. Guhan, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha. Emoticon: Context-aware multimodal emotion recognition using frege's principle. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[66] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, 2017.

[67] T.T. Nguyen, C.M. Nguyen, D.T. Nguyen, and S. Nahavandi. Deep learning for deepfakes creation and detection. *ArXiv*, abs/1909.11573, 2019.

[68] Commission of Evidence-Based Policymaking. The Promise of Evidence-Based Policymaking. `https://www.cep.gov/content/dam/cep/report/cep-final-report.pdf`, 2017.

[69] A. Patra and A. Suresh. Blaze: Blazing fast privacy-preserving machine learning. *arXiv preprint arXiv:2005.09042*, 2020.

[70] S. Peng, L. Chen, C. Gao, and R.J. Tong. Predicting students' attention level with interpretable facial and head dynamic features in an online tutoring system. In *34th AAAI Conference on Artificial Intelligence*, pages 13895–13896, 2020.

[71] R. Poddar, G. Ananthanarayanan, S. Setty, S. Volos, and R.A. Popa. Visor: Privacy-preserving video analytics as a cloud service. In *29th USENIX Security Symposium*, pages 1039–1056, 2020.

[72] Z. Ren, Y. Jae Lee, and M.S. Ryoo. Learning to anonymize faces for privacy preserving action detection. In *European Conference on Computer Vision (ECCV)*, pages 620–636, 2018.

[73] M.S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar. XONN: Xnor-based oblivious deep neural network inference. In *28th USENIX Security Symposium*, pages 1501–1518, 2019.

[74] M.S. Riazi, C. Weinert, O. Tkachenko, E.M. Songhori, T. Schneider, and F. Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Asia Conference on Computer and Communications Security*, pages 707–721, 2018.

[75] B.D. Rouhani, M.S. Riazi, and F. Koushanfar. DeepSecure: Scalable provably-secure deep learning. In *55th Annual Design Automation Conference (DAC)*, 2018.

[76] M.S. Ryoo, B. Rothrock, C. Fleming, and H.J. Yang. Privacy-preserving human activity recognition from extreme low resolution. In *31st AAAI Conference on Artificial Intelligence*, pages 4255–4262, 2017.

[77] National Science and Technology Council. National Privacy Research Strategy. `https://www.nitrd.gov/PUBS/NationalPrivacyResearchStrategy.pdf`, 2016.

[78] A.P. Shah, V. Vaibhav, V. Sharma, M.A. Ismail, J. Girard, and L.P. Morency. Multimodal behavioral markers exploring suicidal intent in social media videos. In *International Conference on Multimodal Interaction*, page 409–413. ACM, 2019.

[79] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.

[80] N. Subramani and D. Rao. Learning efficient representations for fake speech detection, 2020.

[81] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[82] S. Wagh, D. Gupta, and N. Chandran. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 1:24, 2019.

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

[84] H. Wang, Z. Wu, Z. Wang, Z. Wang, and H. Jin. Privacy-preserving deep visual recognition: An adversarial learning framework and a new dataset. *arXiv preprint arXiv:1906.05675*, 2019.

[85] Z.W. Wang, V. Vineet, F. Pittaluga, S.N. Sinha, O. Cossairt, and S. Bing Kang. Privacy-preserving action recognition using coded aperture videos. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[86] Z. Wei, J. Zhang, Z. Lin, J.Y. Lee, N. Balasubramanian, M. Hoai, and D. Samaras. Learning visual emotion representations from web data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[87] J. Wu, L. Wang, Li. Wang, J. Guo, and G. Wu. Learning actor relation graphs for group activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9964–9974, 2019.

[88] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 461–470, 2015.

[89] Zuxuan Wu, Ting Yao, Yanwei Fu, and Yu-Gang Jiang. Deep learning for video classification and captioning. In *Frontiers of multimedia research*, pages 3–29. 2017.

[90] R. Wyden. Wyden pushes for stronger security in collection of personal information. `https://www.wyden.senate.gov/download/20170515-wyden-mpc-letter-to-cep`, 2017.

[91] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[92] R. Yonetani, V. Naresh Boddeti, K.M. Kitani, and Y. Sato. Privacy-preserving visual learning using doubly permuted homomorphic encryption. In *IEEE International Conference on Computer Vision*, pages 2040–2050, 2017.

[93] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[94] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[95] Q. Zhang, C. Wang, H. Wu, C. Xin, and T.V. Phuong. Gelu-net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning.

[96] Q. Zhang, C. Wang, C. Xin, and H. Wu. Cheetah: An ultra-fast, approximation-free, and privacy-preserved neural network framework based on joint obscure linear and nonlinear computations, 2019.

[97] S. Zhao, Y. Ma, Y. Gu, J. Yang, T. Xing, P. Xu, R. Hu, H. Chai, and K. Keutzer. An end-to-end visual-audio attention network for emotion recognition in user-generated videos. In *34th AAAI Conference on Artificial Intelligence*, pages 303–311, 2020.

[98] Y. Zhou, F. Ren, S. Nishide, and X. Kang. Facial sentiment classification based on resnet-18 model. In *2019 International Conference on Electronic Engineering and Informatics (EEI)*, pages 463–466, Nov 2019.