# Contents


**1 Introduction** **3**
1.1 Thesis Structure . . . 4

**2 Background** **4**
2.1 Satellite image analysis . . . 4
2.1.1 Images Format and Resolution . . . 5
2.1.2 Spectral bands use . . . 5
2.1.3 Satellite image preprocessing . . . 6
2.2 Network Background . . . 6
2.2.1 Image Classification . . . 7
2.2.2 Object Detection . . . 7
2.3 Advanced cryptographic techniques for privacy preseving computation . . . 9
2.3.1 Group Anonymity . . . 9
2.3.2 Differential Privacy . . . 9
2.3.3 Multi-Party Computation . . . 10
2.3.4 HE Schemes . . . 10
2.3.5 Overview and Conclusion . . . 11

**3 Literature Analysis** **11**
3.1 State of the Art in Object Detection on Non-Encrypted Satellite Images . . . 11
3.2 Privacy-Preserving Image Classification . . . 12
3.2.1 Channel-Wise Homomorphic Encryption (CHE) . . . 12
3.2.2 Privacy-Preserving CNN Using BFV . . . 12
3.3 Privacy-Preserving Object Detection . . . 12
3.3.1 Privacy-Preserving Object Detection Using Secret Sharing . . . 13
3.3.2 Importance of Lightweight Models in Privacy-Preserving Object Detection . . . 13
3.4 Advantages of the Proposed Solution . . . 13

**4 Proposed Solution** **13**
4.1 Image Preprocessing . . . 13
4.2 Detailed Implementation of HE-Compatible Components . . . 14
4.2.1 Approximation of Non-Linear Operations . . . 14
4.2.2 Object Detection Optimization . . . 14
4.2.3 Simplified Loss Function . . . 15
4.2.4 Ensemble Methods for Improved Detection Accuracy . . . 15
4.2.5 Non Max Suppression and Decoding of Output . . . 16
4.3 Implementation in HE Environment . . . 17
4.3.1 Initialization Steps . . . 17
4.3.2 Encrypt Test Images . . . 17
4.3.3 Perform Predictions Under Encryption . . . 18
4.3.4 Ensemble and Decode Results . . . 18

**5 Experiments** **19**
5.1 Image Classification . . . 19
5.1.1 Dataset . . . 20
5.1.2 Model Architecture . . . 20
5.1.3 Performance . . . 20
5.2 Object Detection . . . 22
5.2.1 Dataset . . . 22
5.2.2 Model Architectures and Performance . . . 22
5.2.3 Experiments under HE . . . 30

**6 Conclusions** **31**
6.1 Further work . . . . . 32

**7 Bibliography and citations** **32**

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Privacy Preserving Object Detection

Tesi di Laurea Magistrale in
Computer Science Engineering - Ingegneria Informatica

**Salvatore Buono, 10600540**

**Advisor:**
Prof. Manuel Roveri

**Co-advisors:**
Luca Colombo
Alessandro Falcetta

**Academic year:**
2023-2024

**Abstract:** In this thesis, we explore the integration of Privacy Preserving techniques within the domain of Deep Learning, with a specific focus on Neural Networks applied to image classification and object detection tasks. While prior research has extensively addressed Privacy Preserving methods in image classification, our work delves into the less explored area of object detection. We focus on the study and adaptation of two prominent architectures, Faster-RCNN and YOLO, to work with homomorphic encryption (HE) schemes, particularly CKKS, ensuring maximal privacy during computation.
Our contributions include the design and implementation of privacy-preserving adaptations of these object detection models, which allow them to perform inference securely under encryption, thereby safeguarding sensitive data. Additionally, we emphasize the optimization of the YOLO architecture to reduce computational overhead, a critical challenge due to the intensive nature of HE operations. Through these optimizations, we demonstrate that it is possible to achieve a more efficient and lightweight model, capable of performing privacy-preserving object detection with reduced inference and training times, without compromising the security of the data.
This work not only advances the field of privacy-preserving deep learning but also provides practical insights into the challenges and potential solutions for implementing secure and efficient object detection systems using HE.

**Key-words:** Privacy Preserving Deep Learning, Satellite Images, Object Detection, Image Classification, Homomorphic Encryption

## 1. Introduction

In today's digital era, safeguarding sensitive data has become a critical priority, particularly in environments where data is processed on a large scale, such as in cloud computing. The challenge lies in ensuring privacy while maintaining the ability to extract valuable insights from this data. Privacy-preserving techniques have therefore emerged as a pivotal area of research, focusing on protecting personal and sensitive information without compromising data utility. This is especially crucial in domains like healthcare, finance, and surveillance, where data security is paramount [11].
Object detection, a fundamental task in computer vision, has been widely adopted across various industries due to its ability to identify and classify objects within images and videos. However, the implementation of object detection algorithms often involves processing sensitive visual data, which can raise significant privacy concerns—particularly when the processing is outsourced to cloud environments.

Cloud computing offers immense computational resources and scalability, making it an ideal platform for handling the intensive workloads required by object detection algorithms. However, this also introduces the risk of exposing sensitive data to potential breaches or unauthorized access. To address these challenges, integrating privacy-preserving mechanisms such as homomorphic encryption (HE) with object detection systems has become an area of growing interest. HE allows computations to be performed on encrypted data, ensuring that the data remains secure throughout the processing pipeline [3].
This thesis focuses on the efficiency of HE for object detection within cloud environments, where the need for privacy preservation is most critical. The use of HE in the cloud enables secure processing of data without compromising the speed and accuracy of object detection algorithms. This efficiency is essential, as cloud-based applications must balance the demands of computational intensity with the need for rapid response times.
A particularly compelling application of this approach is in the processing of satellite images. Satellite imagery is increasingly used for environmental monitoring, disaster management, and military surveillance—domains where both the sensitivity of the data and the need for accurate object detection are high [19]. Cloud computing, combined with privacy-preserving object detection, can offer a robust solution for analyzing these images while safeguarding against unauthorized access to sensitive geographical information.

### 1.1. Thesis Structure

Section 2 introduces the research topic, providing background information and setting the context for the study. It first describes how satellite images dataset features, providing a contextual background of how images are taken from satellite, how can they can be processed in a "black-box" way using neural networks, how can they can be encrypted so that to not expose sensitive information to malicious users.
In Section 3 we review existing literature and state-of-the-art methods relevant to the research problem. This section helps to position the thesis within the broader academic landscape and identify gaps that the current work aims to address.
Section 4 presents the main contributions of the thesis. It details the methodological approach adopted in this research, including the design of algorithms, models, and frameworks developed to solve the research problem. The experiments conducted to validate the proposed methods are described in section 5. It includes details on the experimental setup, datasets used, and performance metrics. The results are then analyzed to demonstrate the effectiveness of the proposed solutions. This section focuses also on experiments conducted under HE using the Pyhelayers library [1]. It discusses how the neural networks were adapted to work with HE, the challenges encountered, and the results obtained. The thesis concludes with a summary of the main findings in Section 6, contributions, and the significance of the work. This section also outlines potential directions for future research, based on the insights gained during the study.

## 2. Background

Satellite imaging [19], neural networks, and cryptographic techniques are key components in modern data analysis, particularly in fields like environmental monitoring, urban planning, and security [30] [35]. The integration of multi-spectral satellite data with advanced neural network architectures has enabled precise image classification and object detection. However, as the use of such data grows, ensuring privacy becomes crucial [10].
This section covers three foundational areas: the characteristics of satellite imagery, including its multispectral nature; neural network architectures optimized for image analysis; and cryptographic schemes like HE and Secure Multi-Party Computation that enable privacy-preserving computations. Understanding these elements is essential for effectively analyzing satellite data while maintaining privacy and security, setting the stage for the methodologies discussed in later sections.

### 2.1. Satellite image analysis

Satellite images have become invaluable assets in various fields, from environmental monitoring to urban planning, due to their ability to capture vast amounts of data across different spectral bands. Among these, multispectral images are particularly important, as they provide rich information by capturing data across multiple wavelengths. This makes them the one of the most useful choice for training neural networks in tasks such as image classification and object detection [19].
In this section, we will explore the key features of satellite images, with a particular focus on multispectral data. These features form the foundation for the datasets that will be employed in the subsequent sections of this thesis. Understanding the characteristics of satellite images, including their spectral, spatial, and temporal

dimensions, is crucial for effectively utilizing them in machine learning models. This analysis will set the stage for developing and implementing neural network architectures tailored to the unique challenges and opportunities presented by satellite imagery.

### 2.1.1 Images Format and Resolution

Landsat and Sentinel are two remote sensing satellite systems that provide high-resolution images of the Earth [19]. Both systems have a number of features, technologies and image types that make them suitable for a variety of application scenarios, including those mentioned in the application. Landsat and Sentinel provide images in GeoTIFF format that embeds geospatial metadata into image files such as aerial photography, satellite imagery, and digitized maps so that they can be used in GIS applications.

| Spatial Resolution | Landsat | Sentinel |
|---|---|---|
| Spatial Resolution | 30 meters | Variable (band-specific) |
| Temporal Resolution | 16 days | 5 days |
| Spectral Resolution | 11 bands | 13 bands |
| Swath width | 185 km | 290 km |
| Types of Images | Multispectral | Mulispectral |
| Size | 5000 * 5000 | Variable |

Table 1: Differences between Landsat and Sentinel

$$Size_{pixel} = \frac{WidthCoveredArea}{SpatialResolution} \times \frac{HeightCoveredArea}{SpatialResolution}$$

For instance, if we consider an area that is 185 kilometers wide and 185 kilometers high, and we want a spatial resolution of 30 meters:

$$Size_{pixel} = \frac{185,000m}{30m} \times \frac{185,000m}{30m} = 6,167 \times 6,167$$

In general, Landsat satellite imagery is widely used for environmental monitoring, while Sentinel imagery is popular for its acquisition frequency and combination of radar and optical data.

### 2.1.2 Spectral bands use

The bands of Sentinel 2 multispectral images are summarized in this table.

| Bands | Central Wavelength (µm) | Resolution (m) |
|---|---|---|
| Band 1 - Coastal Aerosol | 0.443 | 60 |
| Band 2 - Blue | 0.490 | 10 |
| Band 3 - Green | 0.560 | 10 |
| Band 4 - Red | 0.665 | 10 |
| Band 5 - Vegetation Red Edge | 0.705 | 20 |
| Band 6 - Vegetation Red Edge | 0.740 | 20 |
| Band 7 - Vegetation Red Edge | 0.783 | 20 |
| Band 8 - NIR | 0.842 | 10 |
| Band 8A - Narrow NIR | 0.865 | 20 |
| Band 9 - Water Vapor | 0.945 | 60 |
| Band 10 - SWIR Cirrus | 1.375 | 60 |
| Band 11 - SWIR | 1.610 | 20 |
| Band 12 - Shortwave Infrared | 2.190 | 20 |

Table 2: Sentinel 2 bands properties

Sentinel 2's Multi-Spectral instruments can create true colour and false colour images which allows us to bring out information that colours in the visible field cannot detect. An image is called a true colour image when it appears as if a human observer were looking directly at the object. Generally, true colour images are constructed

using primary colour sensors (R,G,B), simulating the actions of the photo-receptor cones of human vision, which are then combined.
Indeed, the easiest way to create a false colour image is to change the mapping order to $(G, B, R) \rightarrow (R, G, B)$ which will change the red colours to green, the green to blue and the blue to red. However, the usefulness of false colour images lies in the fact that they can represent colours not visible to the human eye with RGB images. For example, the combination of bands B8, B4 and B3 maps the images to $(B8, R, G) \rightarrow (R, G, B)$ and thus represents the non-visible spectral band B8 in red. False-colour images are widely used in satellite monitoring as they allow data normally invisible to humans to be highlighted visually.

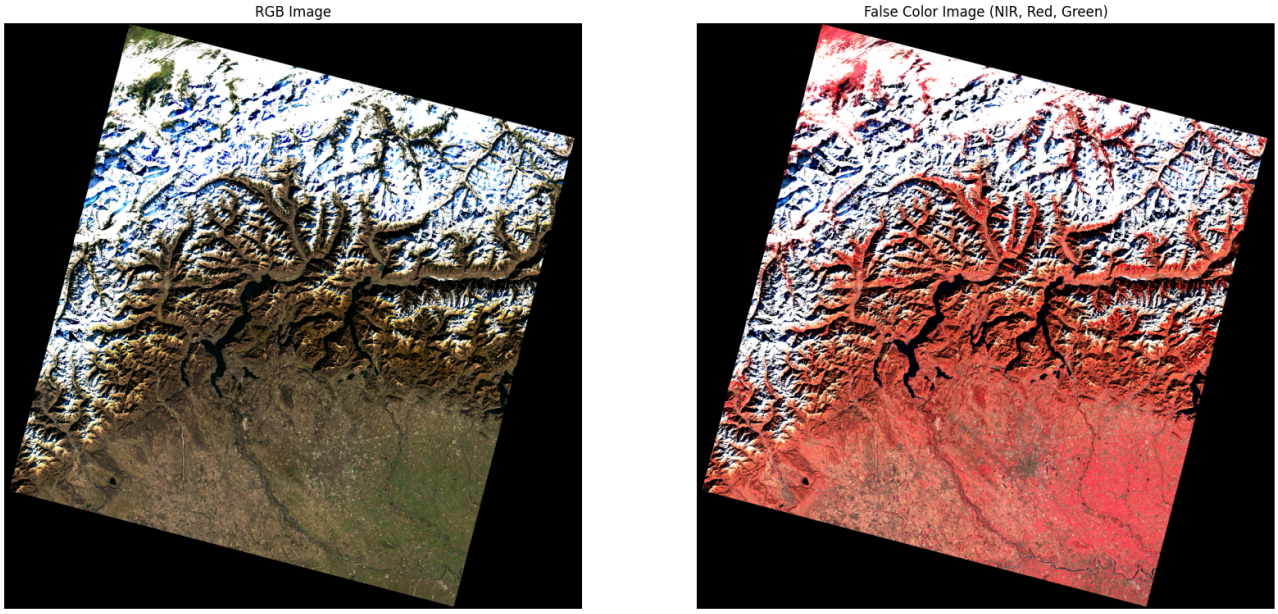



Figure 1: Landsat original images in different bands

### 2.1.3 Satellite image preprocessing

Preprocessing is a crucial step in preparing satellite imagery for analysis, ensuring that the data is clean, accurate, and ready for advanced processing tasks. hese steps collectively enhance the quality and usability of satellite images for detecting illegal activities and environmental monitoring. The main techniques for satellite images prepocessing [29] are:

- **cloud masking**: remove clouds from image in order to ease the detection of illegal operations or illegal sites.
- **Atmospheric Correction**: Satellite images can be affected by atmospheric interference, such as scattering and absorption of light. Atmospheric correction algorithms can be applied to remove these effects and obtain surface reflectance values, making the images more consistent and suitable for analysis.
- **Contrast Enhancement**: Adjusting the contrast of the image can help in highlighting features of interest and improving the visual quality of the data.
- **Dimensionality Reduction**: Some remote sensing applications can benefit from dimensionality reduction techniques such as Principal Component Analysis (PCA) to reduce the number of bands while preserving the most critical information.

## 2.2. Network Background

The success of image classification and object detection tasks heavily relies on the architecture and capabilities of the underlying neural networks. Over the past decade, deep learning has revolutionized these fields, leading to the development of various network architectures, each designed to address specific challenges in image analysis. Understanding these networks is essential, as their performance and suitability can vary significantly depending on the nature of the data and the task at hand.
In this section, we provide an overview of the key neural network architectures relevant to this thesis. We will examine both general-purpose networks and those specifically optimized for handling the complexities of satellite imagery. By exploring the foundational principles, strengths, and limitations of these networks, we aim to build a comprehensive background that will support the proposed methodologies discussed in later sections.

This foundation will help in selecting and adapting the most appropriate networks for the unique requirements of satellite image classification and object detection.

### 2.2.1 Image Classification

Image classification is a fundamental task in computer vision that involves assigning a label or category to an entire image based on its content. The goal is to analyze the visual features of an image and determine which class it belongs to from a predefined set of categories. The network used for image classification are:

- **VGG**: VGGNet [27] is known for its simple yet deep structure with 16-19 convolutional layers. It uses small-sized convolutional filters (3x3) with fixed stride and padding. This simplicity in architecture made it easy to understand and adapt.
- **LeNet**: LeNet [18] architecture consists of five main layers: two convolutional layers for feature extraction, followed by subsampling (pooling) layers, and finally, fully connected layers that lead to the final classification. LeNet was a pioneering model in the application of CNNs and laid the groundwork for many modern deep learning architectures.

Accuracy is a common metric used to evaluate the performance of a model in image classification tasks. It measures the proportion of correctly classified images out of the total number of images.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Prediction} = \frac{TP + TN}{TP + FP + FN + TN}$$

where:

- TP (True Positive): The number of correctly predicted positive classes.
- TN (True Negative): The number of correctly predicted negative classes.
- FP (False Positive): The number of incorrectly predicted positive classes (actually negative but predicted as positive).
- FN (False Negative): The number of incorrectly predicted negative classes (actually positive but predicted as negative).

In image classification task, accuarcy is computed following these steps:

- Prediction: For each image, the model predicts a class label.
- Comparison: The predicted label is compared to the true label (ground truth).
- Counting Correct Predictions: Count how many of these predictions are correct.
- Total Predictions: Count the total number of images classified.
- Calculation: Divide the number of correct predictions by the total number of images.

### 2.2.2 Object Detection

Object detection extends the concept of image classification by not only identifying what objects are present in an image but also determining their locations within the image. This task involves two main components: classification (identifying the type of object) and localization (finding the precise coordinates of the object in the image). Object detection models output bounding boxes that specify the position of each detected object, along with labels indicating what each object is. The used for object detection are:

- **YOLO (You Only Look Once)**: YOLO[23] is an object detection approach that predicts objects in a single pass through the network. It is known for its speed and is widely used in real-time applications like image object detection. YOLO divides an input image into an S $\times$ S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

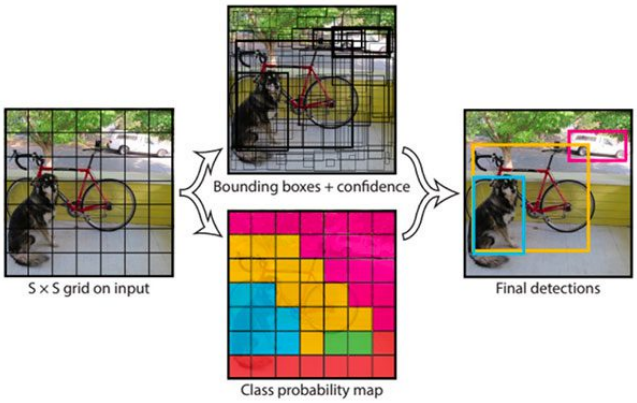



Figure 2: YOLO design architecture

- **Faster R-CNN**: Faster R-CNN [25] is an object detection model that combines a convolutional network for feature extraction with an R-CNN module for actual object detection. It is known for its accuracy and is one of the most common architectures in object detection. The Faster R-CNN (Region-based Convolutional Neural Network) architecture consists of three main components: convolutional layers, a Region Proposal Network (RPN), and a Fast R-CNN detector.
  - Convolutional Layers: The architecture begins with convolutional layers, which are responsible for extracting features from the input image. These layers are trained to identify specific patterns and shapes within the image, such as edges, textures, and other visual elements.
  - Region Proposal Network (RPN): The RPN is a small neural network that slides over the feature map generated by the convolutional layers. Its primary function is to propose regions in the feature map that are likely to contain objects. The RPN predicts whether an object is present in a particular region and also predicts the bounding box for that object.
  - Fast R-CNN Detector: This component consists of a fully connected neural network that performs two tasks: object classification and bounding box regression. It takes the proposed regions from the RPN and predicts the class of the object present in each region, as well as refines the bounding box for the object.

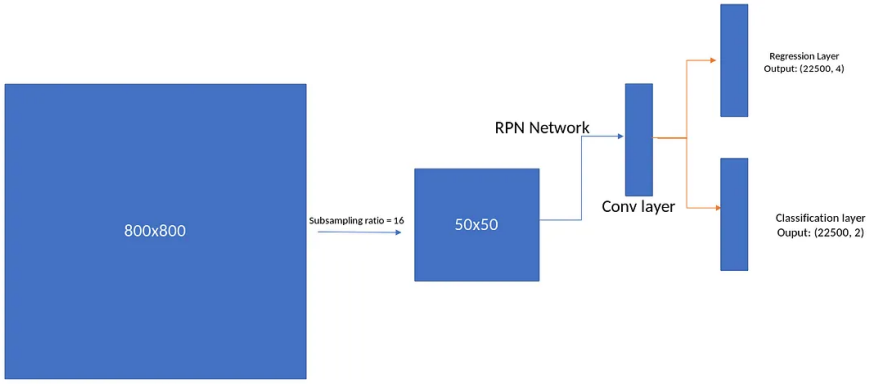



Figure 3: Faster-RCNN architecture

The usual data flow in Faster R-CNN when training the network is as written below:

1. Features Extraction from the image.
2. Creating anchor targets.
3. Locations and objectness score prediction from the RPN network.
4. Taking the top N locations and their objectness scores aka proposal layer
5. Passing these top N locations through Fast R-CNN network and generating locations and cls predictions for each location is suggested in 4.
6. generating proposal targets for each location suggested in 4
7. Using 2 and 3 to calculate *rpn-cls-loss* and *rpn-reg-loss*.
8. using 5 and 6 to calculate *roi-cls-loss* and *roi-reg-loss*.

To evaluate correctly the performance of different object detection the following key concepts should be defined:

- **Intersection Over Union (IoU)**: IoU is a measure of the overlap between two bounding boxes: the

predicted bounding box and the ground truth bounding box (the actual object location).

$$IoU = \frac{Area\ Of\ Overlap}{Area\ Of\ Union}$$

where:

- Area of Overlap: This is the area where the predicted bounding box and the ground truth bounding box intersect.
- Area of Union: This is the total area covered by both bounding boxes, i.e., the combined area of the predicted and ground truth boxes minus their intersection.

The IoU value ranges from 0 to 1:

- $IoU = 1$: Perfect match between the predicted and ground truth boxes.
- $IoU = 0$: No overlap between the predicted and ground truth boxes.

Typically, a prediction is considered a true positive if its IoU with the ground truth is above a certain threshold (e.g., 0.5).

- **Mean Average Precision (mAP)**: mAP is a metric that summarizes the performance of an object detection model across different classes and IoU thresholds. It combines Precision and Recall values to provide a single score that reflects the model's accuracy.
  - Precision: the proportion of true positive detections (correctly predicted bounding boxes) out of all positive detections (all predicted bounding boxes).

$$Precision = \frac{TP}{TP + FP}$$

  - Recall: the proportion of true positive detections out of all possible true objects (ground truth bounding boxes).

$$Recall = \frac{TP}{TP + FN}$$

  Average Precision (AP) is the area under the Precision-Recall curve for a specific class and IoU threshold. Mean Average Precision (mAP) is the mean of APs across all classes and/or IoU thresholds.

## 2.3. Advanced cryptographic techniques for privacy preseving computation

In today's data-driven world, protecting the privacy and security of sensitive information is a major challenge. As organizations collect and process more data, the risk of exposing confidential information increases. To address this, advanced cryptographic techniques are developed to enable privacy-preserving computations, allowing data to be analyzed without compromising individual privacy.
Three main techniques used for privacy-preserving computation are Homomorphic Encryption (HE), Secure Multi-Party Computation (MPC), and Differential Privacy (DP). Each of these approaches tackles different aspects of the challenge, helping to keep data secure during computation.

### 2.3.1 Group Anonymity

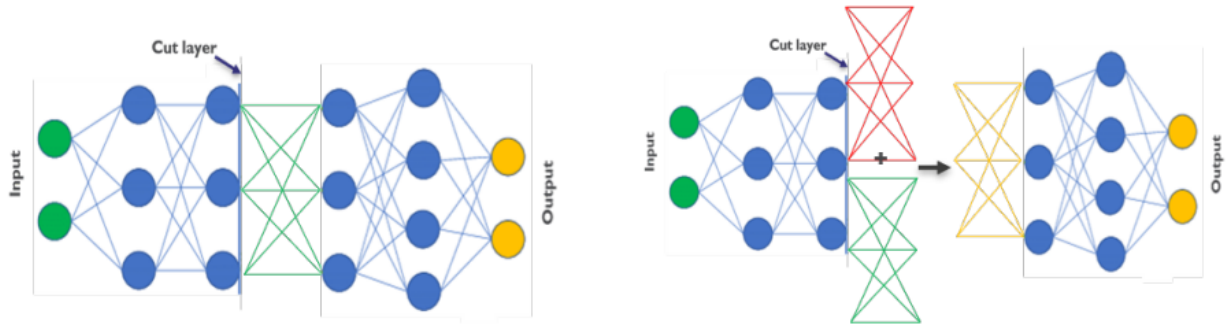
Group anonymity aims to protect the privacy of data in tasks like image classification and object detection, especially when dealing with groups or collections of data. A technique known as the wavelet-based method [13] is often used here, which works by altering certain data patterns, making it difficult to trace information back to specific individuals. The wavelet transform breaks data into different frequency components, which is particularly useful in image processing. By treating a group of related images as a single unit [37], group anonymity helps ensure the privacy of each image.

### 2.3.2 Differential Privacy

Differential privacy is a framework that offers privacy protection for individuals in a dataset while still allowing useful patterns to be discovered. In machine learning tasks like image classification, this often involves injecting noise into the training process. For example, techniques like Differentially Private Stochastic Gradient Descent (DP-SGD) [26] [6] modify the standard gradient descent algorithm to include noise, which limits the amount of private information that can be inferred from any single update. While adding more noise strengthens privacy, it can also reduce the accuracy of the final model.

### 2.3.3 Multi-Party Computation

MPC is particularly useful when multiple parties need to collaborate on tasks like satellite image classification without sharing their private data. For instance, organizations with access to different satellite images may want to jointly analyze land use changes, but privacy concerns prevent them from sharing their data directly. MPC allows them to collaborate by computing the desired results without revealing any private data. Each party performs computations on its own data and shares only computed results, which are combined to update the model without exposing sensitive information [4] [22] [5].
MPC is especially beneficial for tasks like training deep learning models on data distributed across different organizations, ensuring that privacy is maintained throughout the process.




(a) Splitting YOLO on device1 (b) Aggregating local and received features in device1

Figure 4: Multi-Party Computation with YOLO architectures

This way, MPC enables privacy-preserving collaborative learning and prediction, which can be very beneficial in fields like remote sensing where data privacy and security are of great interest.

### 2.3.4 HE Schemes

HE is a type of encryption that allows computations to be performed on encrypted data without first decrypting it. This means that the data remains in an encrypted form throughout the computation process, and the output, when decrypted, is identical to the result that would have been obtained if the operations were performed on the unencrypted data. [3] Types of HE are:

- FHE: an unlimited number of both addition and multiplication operations.
- Somewhat HE: a limited number of either addition or multiplication operations, but not both.
- Partially HE: only one addition or multiplication operation, but not both.
- Leveled HE: relinearization and modulus reduction performed on a limited number of operations.

Fully homomorphic encryption (**FHE**) [10] allows a worker to receive encrypted data and perform arbitrarily complex dynamically chosen computations on that data while it remains encrypted, despite not having the secret decryption key.
In an encryption scheme, a message (aka plaintext) is encrypted with a key to produce a ciphertext.
CKKS is a type of HE scheme designed to allow computations on encrypted data, but unlike other schemes like BFV, it works with approximate arithmetic. This means that CKKS can handle numbers like decimals and real numbers, but the results of computations might not be exact—they're close, but not perfectly accurate. This slight inaccuracy is acceptable for many applications where results don't need to be perfectly precise, such as when working with real numbers in tasks like image processing or machine learning.
One of the unique aspects of CKKS is that it uses complex numbers to perform operations on both real and complex values. This is what enables CKKS to handle approximate arithmetic efficiently. Before performing any computation, the data (a vector of values) is encoded into a polynomial and then encrypted using a public key. Once encrypted, CKKS allows various operations like addition, multiplication, and rotation to be performed directly on the encrypted data.
These key operations and concepts are influenced by several important parameters:

- The **number of slots** determines how much data can be processed simultaneously.
- The integer and fractional part **precision** control how accurately whole numbers and decimals are represented.
- The **multiplication depth** limits how many consecutive multiplications can be done before the encrypted data becomes too noisy to decrypt accurately.
- The **rotation keys** and their **offsets** enable more complex transformations needed for advanced computations.

### 2.3.5 Overview and Conclusion

| | Advantages | Disadvantages |
|---|---|---|
| Homomorphic Encryption | Enable fully computation on encrypted data | Requires application modification and slow performances |
| Differential Privacy | Protects privacy adding noise | May introduce inaccuracies due to noise |
| Group Anonymity | Good perfomances | Lower level of privacy |
| Multi Party Computation | Allows computations on encrypted data from multiple sources | Computational complexity, data diversity and scalability |

Table 3: Differences between advanced cryptographic techniques

From analysis done in previous sections and in table 3, the method used in this thesis to preserve privacy is HE, since the main advantage lies in its ability to perform computations directly on encrypted data without needing to decrypt it first, it does not send information across the nodes, since it can be computed only in a unique remote node and decrypted results have the same accuracy as if they are computed only in clear [9] [10]. This feature is particularly useful when dealing with sensitive data that needs to be processed by an untrusted third party, such as a remote server.

# 3. Literature Analysis

Satellite image classification and object detection play pivotal roles in remote sensing, providing essential tools for applications ranging from environmental monitoring to public safety. With the growing concern over data privacy, many researchers have focused on developing privacy-preserving techniques that maintain the utility of these applications without compromising sensitive information. This section is divided into two subsections: one covering privacy-preserving image classification and the other focusing on privacy-preserving object detection, with a particular emphasis on lightweight models.

## 3.1. State of the Art in Object Detection on Non-Encrypted Satellite Images

Neural networks play a crucial role in analyzing satellite images, especially in detecting illegal activities and monitoring environmental changes. In the context of object detection using non-encrypted satellite images, advanced neural networks and image analysis techniques are applied to various scenarios, such as identifying illegal mining, spotting illegal dumping, and monitoring deforestation.
For detecting illegal mining [16], a combination of unsupervised and supervised methods is often employed. The process begins by clustering satellite data into regions with similar spectral characteristics, followed by supervised classification techniques like maximum likelihood classification to create accurate maps of land cover changes. This approach, validated with field data and satellite images, helps identify areas impacted by mining activities.
Neural networks also play a significant role in detecting illegal dumping using deep learning models such as Single Shot Detector (SSD) [35]. SSD identifies objects by assigning bounding boxes at multiple scales, refining these detections with non-maximum suppression to keep only the most relevant results. Additionally, Autoencoders detect anomalies in satellite images by learning normal patterns and identifying deviations through reconstruction errors. These models create binary segmentation masks, which are then fine-tuned through post-processing to enhance accuracy.
When it comes to deforestation monitoring, neural networks are used to track changes in forest cover by analyzing large-scale satellite images [30]. These models can detect areas of illegal logging and forest degradation by comparing temporal data, making them essential for environmental conservation efforts.

While these methods are highly accurate in detecting illegal activities and monitoring environmental changes, they come with significant complexity. The models require a large amount of computational power and are often difficult to implement due to the sophisticated algorithms involved. Techniques like temporal analysis, anomaly detection, and bounding box refinement demand precise tuning and extensive post-processing, which adds to the overall difficulty of deploying these systems at scale.

## 3.2. Privacy-Preserving Image Classification

In the realm of privacy-preserving image classification, HE has emerged as a key technique. Two significant contributions in this area are Channel-Wise Homomorphic Encryption (CHE) [34] and the privacy-preserving CNN methodology using the BFV scheme [9].

### 3.2.1 Channel-Wise Homomorphic Encryption (CHE)

One of the most recent approach to enable neural network to work behind HE is the Channel-Wise Homomorphic Encryption (CHE) [34] which tackles the challenge of balancing privacy and computational efficiency in deep learning, especially for image classification. Instead of encrypting each pixel, CHE encrypts data at the channel level, which reduces the typical computational load associated with HE. By converting each image channel into a vector and encrypting it into a single ciphertext, CHE speeds up processing through SIMD (Single Instruction Multiple Data) operations.
In CHE, convolution operations are broken down into two key steps: rotation and accumulation, and mask operations. This structure has unique advantages, allowing the system to perform convolutions on encrypted data more efficiently, which in turn reduces model latency. CHE also introduces two configurations, CBA (Convolution-BN-Activation) and CAB (Convolution-Activation-BN) to help find a balance between speed and accuracy.
However, despite these benefits, CHE has some limitations. The most important is that it doesn't support batch processing, which can greatly affect throughput, especially when working with large datasets. This limitation makes CHE less ideal for more complex or large-scale image classification tasks, where batch processing is important for efficiency.

### 3.2.2 Privacy-Preserving CNN Using BFV

Another type of methodology used in privacy preserving image classification [9] aims to create a privacy-preserving version of CNNs using the BFV HE scheme. This approach focuses on adapting CNN layers to be compliant with the BFV scheme, which only supports operation which can be approximated with polynomials. The study addresses key challenges such as replacing operations that are incompatible with BFV (e.g., maximum pooling, normalization layers, and nonlinear activation functions like ReLU) with operations that can be performed on encrypted data.
For instance, maximum pooling layers are substituted with average pooling layers, and batch normalization is employed instead of standard normalization to maintain compatibility with encrypted data. Nonlinear activation functions, which require comparison operations not supported by BFV, are replaced with polynomial approximations like the square function.
The methodology proceeds through three stages: model approximation, encoding, and validation. The model is approximated to fit the constraints of HE, encoded according to the BFV scheme, and then validated to ensure that the noise budget (NB) is sufficient, i.e the amount of noise which can be tolerated during computation on encrypted data, and that the loss of accuracy remains within acceptable limits. The study highlights the need for careful parameter tuning, particularly with respect to the noise budget and polynomial degree, to maintain accuracy while minimizing computational overhead.
Despite its contributions, this approach faces limitations such as increased computational overhead and potential accuracy loss due to approximations and parameter constraints. Furthermore, the complexity of parameter tuning and the overhead associated with maintaining an adequate noise budget make this approach challenging to implement in scenarios requiring high scalability and efficiency.

## 3.3. Privacy-Preserving Object Detection

Object detection in privacy-sensitive environments creates unique challenges, particularly when ensuring that sensitive data remains secure throughout the detection process. Several approaches have been proposed, utilizing techniques such as Secret Sharing and lightweight models to address these challenges.

### 3.3.1 Privacy-Preserving Object Detection Using Secret Sharing

The Secure YOLOv3-SPP framework [24] [38] makes use of Secret Sharing for privacy-preserving object detection in Connected Autonomous Vehicles (CAVs). In this framework, images captured by onboard sensors are split into secret shares, which are then distributed to edge servers. Each server performs detection tasks on its share, ensuring that no single server can reconstruct the entire image, thus maintaining privacy.
However, the Secure YOLOv3-SPP framework is not without its challenges. The process of splitting images into shares and performing secure computations across multiple servers introduces significant communication overhead. This overhead can be particularly problematic in scenarios with limited bandwidth or high network latency, potentially limiting the framework's applicability in real-time systems.

### 3.3.2 Importance of Lightweight Models in Privacy-Preserving Object Detection

Given the high computational costs associated with HE and Secret Sharing, lightweight models have become increasingly important in privacy-preserving object detection. Models like TinyissimoYOLO [21] and Tiny-YOLOv3 [24] are designed to run efficiently on low-power devices, making them suitable for environments with limited computational resources.
TinyissimoYOLO [21], for instance, is optimized to operate with just 422k parameters, allowing deployment on microcontrollers with less than 0.5 MB of memory for CNN weights. These models achieve real-time object detection with minimal computational resources, which is crucial for applications where power efficiency and speed are paramount.
In addition to Tiny models, the FOMO (Faster Objects, More Objects) [14] approach represents a significant shift in object detection methodologies. FOMO simplifies the detection process by focusing on the identification of object centroids rather than full bounding boxes. This reduces the computational complexity and makes it feasible to deploy object detection models on devices with very limited computational capabilities.
While these lightweight models offer significant advantages in terms of efficiency, they are not without trade-offs. The simplifications necessary to achieve lightweight performance can lead to reduced accuracy, particularly in detecting more complex objects or in scenarios requiring high precision. However, the integration of these models with privacy-preserving techniques could offer a viable solution for balancing performance and privacy in constrained environments.

## 3.4. Advantages of the Proposed Solution

Unlike the CHE method, which operates at the channel level and lacks batch processing capabilities, the proposed solution is designed to handle more complex tasks, such as full object detection, with batch support. This feature directly addresses the throughput limitations of CHE [34], making the solution better suited for real-time applications in remote sensing environments where latency is critical.
Furthermore, while lightweight models like TinyissimoYOLO [21] focus on reducing computational overhead to fit constrained environments, they often sacrifice accuracy, especially in handling complex object detection tasks, the proposed solution optimizes network architectures to balance computational efficiency and privacy-preserving object detection.

# 4. Proposed Solution

The proposed solution introduces a new methodology for adapting traditional neural network architectures to operate efficiently within a HE environment, particularly tailored for object detection tasks. The core objective is to modify standard network operations to be compatible with the constraints of HE, such as limited support for non-linear operations, while preserving the accuracy and computational efficiency of the original models. The primary contribution of this work is the systematic transformation of a standard object detection network into an HE-compatible version, with specific adjustments to activation functions, pooling layers, and loss functions, ensuring efficient encrypted data processing.

## 4.1. Image Preprocessing

The preprocessing step is fundamental beacuse it prepares image data for neural networks by making it consistent, clean, and suitable for the model's requirements. It also plays a crucial role in enhancing the network's ability to learn meaningful features and patterns from the data, leading to improved model performance and generalization.

- **data augmentation**: is a common preprocessing technique used to increase the diversity of the training dataset. It involves applying random transformations to images, such as rotations, flips, and small translations. Data augmentation helps the model generalize better and become more robust to variations in input data.
- **Normalization**: Images often come in various formats, sizes, and color spaces. Preprocessing helps to standardize the input data. Normalization typically involves scaling pixel values to a common range (e.g., $[0, 1]$ or $[-1, 1]$) to ensure that all images have the same magnitude. This ensures that the neural network can learn effectively from the data without being biased by different scales.
- **Resizing**: Convolutional neural networks (CNNs) typically require fixed-size input. Images in a dataset may come in different sizes, so preprocessing involves resizing them to a consistent size that the neural network can handle. Common sizes include 88x88 or 256x256 pixels for many architectures.
- **Validation set**: it is important to split dataset in 3 disjoint sets: training, validation and testing in order to overcome overfitting and have a better estimation of true performance of our model. A valid split can be 70/20/10.

## 4.2. Detailed Implementation of HE-Compatible Components

The following sections describe the specific implementations of the HE-compatible components in the proposed solution, focusing on activations, loss functions, and post-processing.

### 4.2.1 Approximation of Non-Linear Operations

Standard neural networks often include non-linear components like ReLU activations and max pooling layers, which are incompatible with the linear operations supported by HE. To adapt these:

- **ReLU Activation:** The ReLU function is approximated using a polynomial function, such as $f(x) = x + x^2$. This quadratic approximation maintains the basic behavior of ReLU—allowing positive inputs to pass through unchanged while suppressing negative inputs—while ensuring that the function remains computable within the HE environment [2].

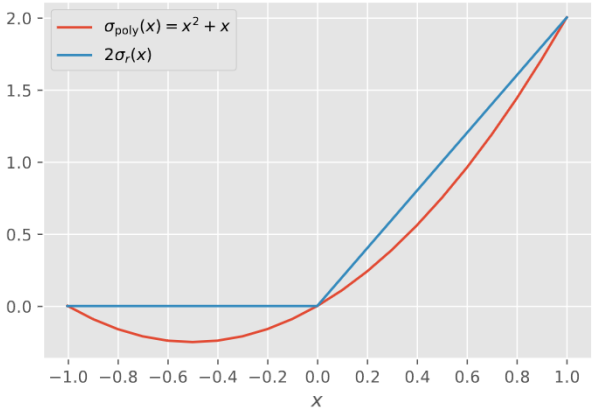



Figure 5: Approximation of ReLu with polynomials

- **Pooling Layers:** Max pooling, a common operation in convolutional networks, is replaced with average pooling, as it relies solely on addition and division, which are supported by HE [1]. This substitution simplifies the network while still enabling the reduction of spatial dimensions in feature maps.

These approximations are crucial in maintaining the integrity of the neural network's performance while ensuring compatibility with the constraints of HE.

### 4.2.2 Object Detection Optimization

A critical aspect of adapting standard neural networks for privacy-preserving object detection under HE is the object detection architecture.

One of the critical challenges in HE is managing the multiplicative depth of operations, which directly impacts the noise budget and the feasibility of decryption. The solution involves minimizing the multiplicative depth by reducing the number of layers and simplifying operations wherever possible [9]. For instance, the use of fewer convolutional layers and reducing the number of operation during the forward pass of the network help in maintaining a lower multiplicative depth.

In traditional object detection tasks, the loss function typically involves calculating the difference between predicted and actual bounding boxes, which includes coordinates for the center, width, and height of the box, as well as classification scores. However, under HE, this task is computationally expensive due to the need for operation needed to computes coordinates and sizes of bounding boxes. The proposed solution simplifies this by focusing on the prediction of object centers and euclidean distance between them rather than considering the full bounding boxes like in FOMO approach [14]. This reduces the complexity of the computation while still allowing the model to localize objects effectively.

### 4.2.3 Simplified Loss Function

The loss function used in traditional object detection models is adapted to focus on predicting the coordinates of object centers rather than full bounding boxes. The redesigned loss function for the YOLO[23] architecture is as follows:

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&\quad + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{\text{obj}} \left( C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{I}_{i}^{\text{obj}} \sum_{c \in \text{ classes}} \left( p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}$$

Where:

- $S$ Grid size (default is 4). The input image is divided into an $S \times S$ grid.
- $B$: Number of centers per grid cell.
- $\mathbb{I}_{ij}^{\text{obj}} = 1$ if object appears in cell $i$ and is assigned to center $j$, otherwise 0.
- $\mathbb{I}_{ij}^{\text{noobj}} = 1$ if no object appears in cell $i$ for center $j$, otherwise 0.
- $(x_i, y_i)$ are the ground truth centers coordinates.
- $(\hat{x}_i, \hat{y}_i)$ are the predicted centers coordinates.
- $C_i$ is the confidence score that the predicted box $j$ contains an object.
- $p_i(c)$ is the class probability for class $c$.
- $\lambda_{\text{coord}}$ is a weight factor for the bounding box coordinate loss.
- $\lambda_{\text{noobj}}$ is a weight factor for the no-object confidence loss.

### 4.2.4 Ensemble Methods for Improved Detection Accuracy

To further enhance detection accuracy, especially for smaller objects, ensemble methods are employed. The ensemble process involves combining the predictions of multiple models, each trained on different dataset initializations. Two approaches are explored:

- **Confidence-Based Maximum Selection:** Among the $n$ models in the ensemble, the final output is determined by selecting the prediction with the highest confidence score. While this approach is accurate, it requires decrypting the outputs before combining them, which can decrease security levels.
- **Weighted Averaging:** The ensemble output is calculated as a weighted average of the predictions from each model. This approach, which can be performed within the HE framework, shifts the computational load to the cloud, allowing the user to decrypt only a single, aggregated output. The weights are determined based on the validation error of each model, ensuring that more accurate models contribute more significantly to the final prediction. An example of this approach with 2 models can be the following:
  - Model 1: obataines a final validation error $v_1$ and on input $x$ returns as prediction $p_1$ then

$$w_1 = (1 - \frac{v_1}{v_1 + v_2})$$

  - Model 2: obataines a final validation error $v_2$ and on input $x$ returns as prediction $p_2$ then

$$w_2 = (1 - \frac{v_2}{v_1 + v_2})$$

  Then the final prediction on $x$ is:

$$y = w_1 \cdot p_1 + w_2 \cdot p_2$$

- **Weighted Boxes Fusion (WBF):** For bounding box predictions, a Weighted Boxes Fusion function is employed. This technique takes the predictions from different models and combines them using a weighted average based on the confidence scores. The WBF method enhances detection accuracy by ensuring that the final bounding boxes are more accurate representations of the object locations.

$$\text{WBF} = \frac{\sum_{i=1}^{N} w_i \cdot \mathbf{b}_i}{\sum_{i=1}^{N} w_i}$$

The ensemble methods significantly improve the overall robustness and accuracy of the HE-adapted object detection network, particularly in scenarios with multiple or small objects.

### 4.2.5 Non Max Suppression and Decoding of Output

After decryption, the relevant information from the predictions is extracted, particularly the center coordinates of detected objects relative to the image dimensions, not just the cells into which the images are split (as per the YOLO framework [23]).

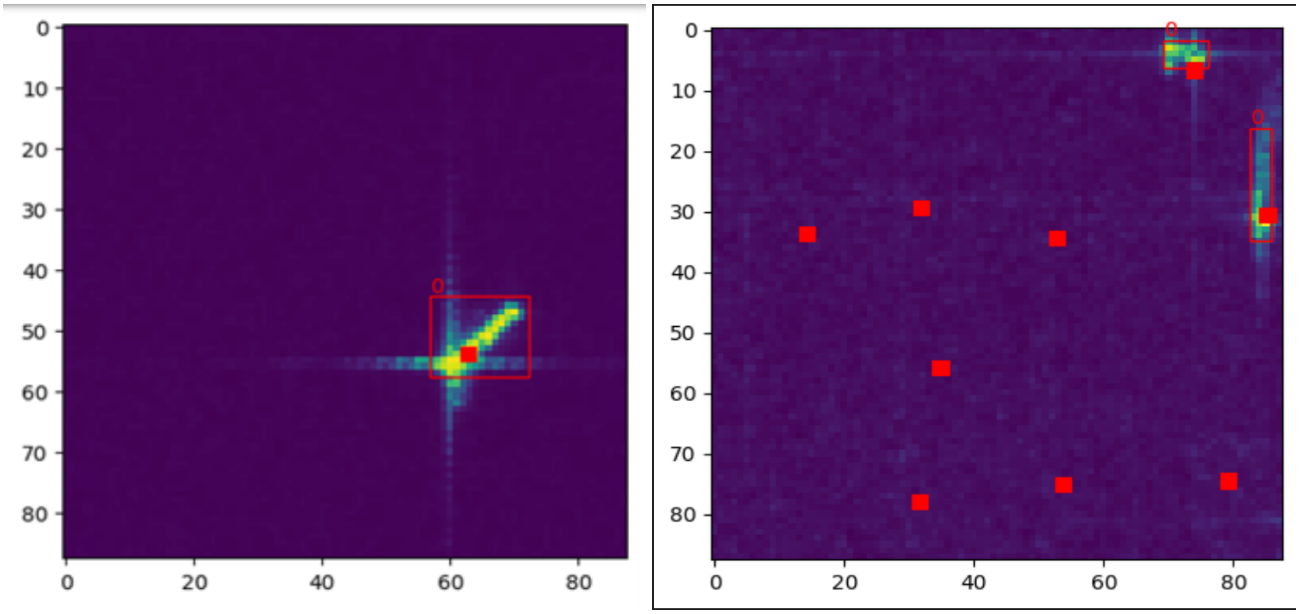
Algorithm 1 Get Centers from model output

1: Set model to evaluation mode
2: **for** x, labels in loader **do**
3: $predictions = model(x)$
4: $predcenters = convertcellcenters(predictions)$
5: $nmscenters = nms(predcenters)$
6: **end for**

Algorithm 2 Converts the predictions of a model to coordinates of centers within a grid

1: Reshape predictions to shape $(batchsize, S, S, C + B * 3)$
2: select x and y from $predictions[..., C + 1 : C + 3]$
3: Calculate the x and y coordinates by adding the grid indices to the center predictions and scaling by 1/S
4: Extract the confidence scores from the predictions
5: Concatenate predicted class, best confidence, and converted centers to form the final predictions

The NMS (Non-Max-Suppression) technique is adapted to work with center points rather than bounding boxes. By focusing on the most accurate center points, NMS helps minimize false positives and ensures that each object is represented by a single, precise center point. The adapted NMS filters out redundant center predictions by considering only those with the highest confidence scores, thus improving detection accuracy.



(a) Output with NMS activated

(b) Output with NMS disabled

Figure 6: NMS effects on output

## 4.3. Implementation in HE Environment

In implementing object detection within a HE environment, the workflow must address several critical steps to ensure that both the data and the neural network remain encrypted throughout the process. This is essential for maintaining privacy and ensuring that sensitive data is never exposed. The CKKS encryption scheme is particularly suited for this task due to its support for approximate arithmetic operations, making it ideal for neural network operations where some degree of imprecision is acceptable.

### 4.3.1 Initialization Steps

The process begins with selecting and initializing the CKKS scheme. This encryption method is chosen specifically for its ability to handle approximate arithmetic, which is crucial for neural networks that rely on continuous and probabilistic data rather than exact values [10]. The system is configured by automatically adjusting key parameters like the polynomial modulus and the number of ciphertext slots thanks to Pyhelibrary [1]. These optimizations are crucial as they balance the computational load while ensuring that the noise levels remain within a manageable range. Noise is an inherent aspect of HE, and controlling it is essential for producing results that can be accurately decrypted later [9]. The batch size is another parameter that must be set carefully, as it affects the number of samples processed simultaneously and can influence both performance and accuracy.

Algorithm 3 Initialize CKKS scheme

```
function initialize_he_environment()
    Select CKKS encryption scheme
    pyhelayers.optimizeParameters()
    Set batch size for processing
    return ckks_scheme
```

### 4.3.2 Encrypt Test Images

Once the system is initialized, the test images are prepared for encryption. This process is managed by the Pyhelayers library [1], which abstracts much of the complexity involved in HE. This ensures that the encryption format aligns with what was used during training or plain inference, maintaining consistency. The encryption transforms these values into ciphertexts that are unreadable by any unauthorized parties. These encrypted data objects are now ready for processing by the neural network without any risk of revealing sensitive information.

Algorithm 4 Encrypt Test Images

1: Encrypt test images using Pyhelayers
2: send public key to server
3: **return** encrypted_images

### 4.3.3 Perform Predictions Under Encryption

With the data encrypted, the inference process is ready to begin. At this point, the encrypted data is sent to an untrusted server, which performs the computation. Crucially, because the data is encrypted, the server has no access to the actual content of the images, or the predictions it generates. The server is simply performing mathematical operations on encrypted values, ensuring that all sensitive information remains hidden throughout the entire process. This ability to perform operations on encrypted data without ever decrypting it is the main advantage of privacy preservation in HE.

Algorithm 5 Perform Predictions Under Encryption

1: Encrypt model using enviromnent created with client public key
2: Perform computations on encrypted data
3: **return** encrypted_predictions

### 4.3.4 Ensemble and Decode Results

Before fully finalizing the output, ensemble techniques can be applied if necessary. Different predictions from multiple models can be combined to improve accuracy. This step helps mitigate errors that might arise from any one model's prediction.

Algorithm 6 Apply Ensemble Methods

1: Combine predictions from multiple models using ensemble techniques
2: **return** ensembled_predictions

Once the inference process is complete, the encrypted results are returned to the trusted client for decryption. This is where the results, still in encrypted form, are transformed back into usable data. The decrypted results provide the object detection predictions.
Finally, NMS algorithm is applied to the predictions. NMS ensures that redundant bounding boxes are eliminated, so each object in the image is represented by a single, accurate bounding box. Once this refinement is complete, the final predictions can be visualized and confirmed for accuracy.

Algorithm 7 Decrypt and Process Results

1: Decrypt predictions on trusted client
2: Apply NMS
3: **return** decrypted_predictions

This workflow highlights the integration of HE into the object detection process, ensuring that data privacy is maintained without compromising the accuracy or efficiency of the neural network [15].

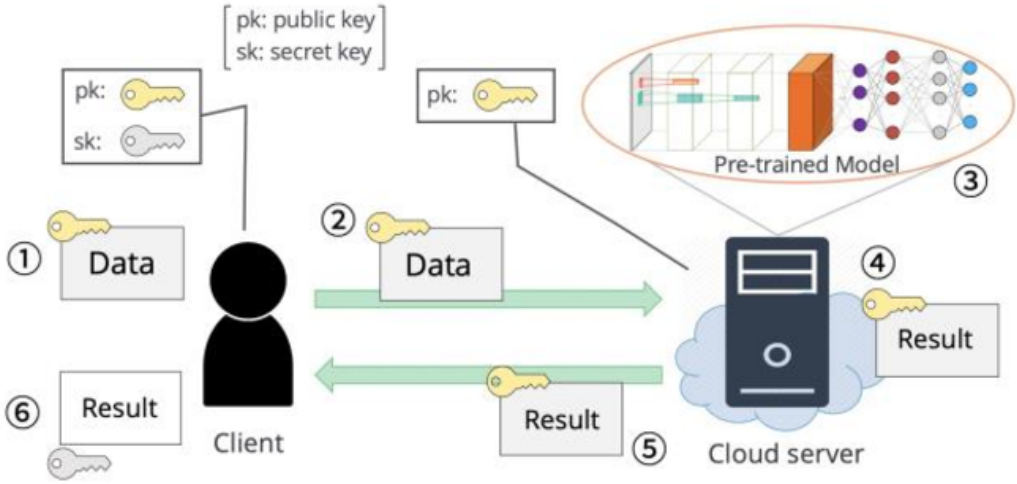



Figure 7: HE workflow

**Algorithm 8** Main Workflow

1: Initialize HE environment: he_env = initialize_he_environment()
2: Encrypt data: encrypted_images = encrypt_data(test_images)
3: Load encrypted model: encrypted_model = load_encrypted_model(he_env)
4: Perform inference: encrypted_predictions = perform_encrypted_inference(encrypted_images, encrypted_model)
5: Decrypt results: decrypted_predictions = decrypt_and_process_results(encrypted_predictions)
6: Apply ensemble (optional): final_predictions = apply_ensemble_methods(decrypted_predictions)
7: Apply NMS: final_predictions = apply_nms(final_predictions)
8: Visualize final predictions

## 5. Experiments

In the initial implementation and experimentation phase, the HE-adapted object detection network was evaluated for its ability to operate on encrypted data while maintaining accuracy comparable to its non-HE counterparts. The experiments revealed that by focusing on center predictions and utilizing polynomial approximations for non-linear functions, the network could effectively process encrypted images and produce accurate detection results.

### 5.1. Image Classification

The first set of experiments focuses on image classification, a task that, while simpler than object detection, serves as a critical starting point for understanding the implementation of CNN within HE schemes. These experiments utilize multispectral images from the original Landsat and Sentinel datasets. For the Landsat dataset, hyperspectral images were acquired using the Earth Explorer API, specifically targeting the Milan area. The images were cropped to various sizes to assess the impact of image dimensions on inference time, a key factor when considering the computational overhead introduced by HE.
While image classification under HE has been previously studied in the literature [9], this research aims to evaluate the performance of CNNs specifically designed to be compatible with HE schemes. By analyzing different CNN configurations, the experiments seek to provide insights into optimizing neural network architectures for encrypted data processing, balancing accuracy and efficiency under the constraints of HE.
The original Landsat and Sentinel datasets lack labeled data and well-defined classes, prompting a search for a more detailed dataset that includes Landsat or Sentinel images.

### 5.1.1 Dataset

The EuroSAT dataset [12] consists of 27,000 labeled Sentinel-2 satellite images, each measuring 64x64 pixels across 13 spectral bands. The dataset spans 10 distinct land use and land cover classes, captured from various geographic locations across Europe. It is specifically designed for training and evaluating machine learning models in tasks such as land use classification, urban planning, agriculture monitoring, and environmental studies. The dataset includes the following 10 classes:

1. Annual Crop
2. Forest
3. Herbaceous Vegetation
4. Highway
5. Industrial
6. Pasture
7. Permanent Crop
8. Residential
9. River
10. Sea/Lake

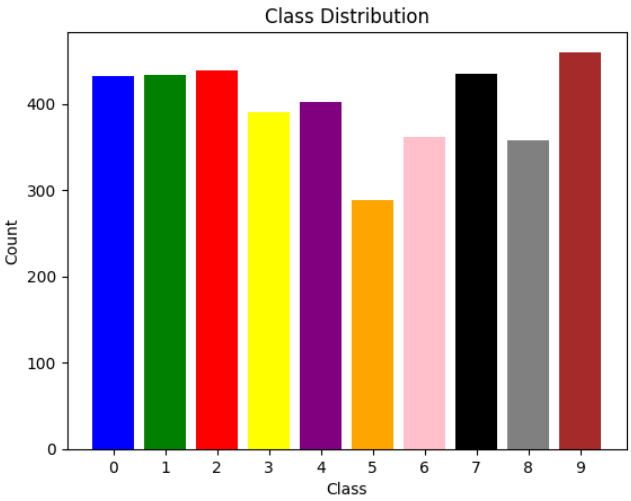



Figure 8: Class Balance

### 5.1.2 Model Architecture

To ensure compatibility with HE schemes, the model employs a linear activation function, specifically the **SquareActivation**. Given the hardware limitations and the added complexity introduced by HE, the model architecture is deliberately kept simple, balancing performance with feasibility.
The model contains a total of 515,146 parameters and has a size of 1.97 MB.

### 5.1.3 Performance

The highest accuracy scores on the EuroSAT dataset were achieved with the following CNN architectures:

- ResNet50: A 50-layer network utilizing residual blocks, enabling better learning by preserving information from earlier layers. Accuracy: 0.98.
- GoogLeNet: A 22-layer network with inception blocks, reducing computational complexity and the number of parameters. Accuracy: 0.96.
- Simplified CNN: A model with 2 convolutional layers, achieving an accuracy of 0.86.

| Component Type | Quantity | Characteristics |
|---|---|---|
| Input Layer | 1 | Input shape: (64, 64, 13), representing a 64x64 multispectral image with 13 channels. |
| Convolutional Layer | 2 | - First Conv2D: 16 filters, kernel size of 5x5, stride of 2.<br>- Second Conv2D: 32 filters, kernel size of 5x5, stride of 1. |
| Square Activation | 3 | Applies a square operation to the inputs. Used after each Conv2D and Dense layer. |
| Average Pooling | 1 | Pool size: 2x2, reduces spatial dimensions by half. |
| Batch Normalization | 1 | Normalizes activations to stabilize and accelerate training. Applied after the first pooling layer. |
| Flatten Layer | 1 | Converts the 3D tensor into a 1D vector for the fully connected layers. |
| Dense Layers | 2 | - First Dense: 128 units, connected to the Flatten layer.<br>- Second Dense: 10 units, serving as the output layer for classification. |
| Loss Function | 1 | *binary_cross_entropy*, This function measures the dissimilarity between model-predicted values and actual (labeled) values for data. |
| Optimizer | 1 | Adam optimizer, known for its adaptive learning rate and efficiency in handling large datasets. |

Table 4: Architecture of the image classification network

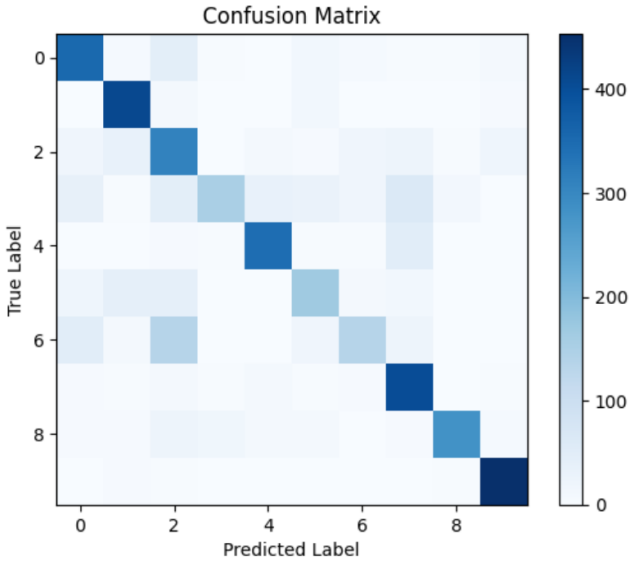



Figure 9: Confusion Matrix

The model architecture compatible with HE achieved an accuracy of 0.75, primarily due to the absence of the ReLU activation function, which is known to enhance performance stability and the use of few convolutional layer, which decrease the ability to learn complex pattern between data.

## 5.2. Object Detection

This section presents a comprehensive analysis of the results derived from implementing the proposed solution, thoroughly evaluating each step outlined in Section 4. By beginning with the object detection network, we systematically explore the impact of incremental modifications on overall performance. We explore why even small changes can greatly improve performance, offering a clear explanation of the factors that lead to these improvements. Furthermore, we discuss how these insights can be leveraged to fine-tune the object detection task, optimizing its efficiency and accuracy, particularly in the context of a HE environment. This analysis not only demonstrates the practical implications of the proposed solution but also offers a deeper understanding of the intricate balance between model adjustments and real-world performance, providing a roadmap for future applications and improvements.

### 5.2.1 Dataset

Object detection task consists of idetifying the position and the type of a specific object inside an image. It is relatevely more complex than image classification, so it needs havier networks. Different model architectures are deepen to transform a classic version of object detection network into a one compatible with FHE schemes. The datasets FAIR1M [28], DOTA [8] [33] [7], DIOR [20], xView [17], and AI-TOD [31] are all significant resources for object detection in satellite and aerial imagery, each with unique characteristics and contributions to the field (Table 5).
The dataset chosen for first test is DIOR, since it provides a wider ranges of classes, a discrete number of samples and a fixed image size, which can be useful since neural networks takes as input an image of fixed size, so deleting the rescaling or up-scaling which can decrease the quality of image.

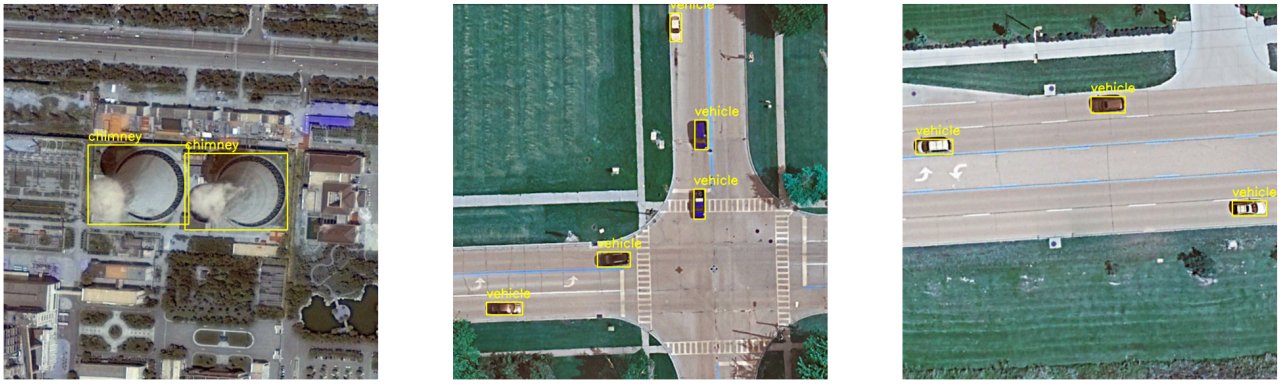



Figure 10: Examples of DIOR target bounding boxes with relative classes

After an in-depth analysis of the performances obtained on this dataset (Figure 13), to obtain the lightest possible network which can reach discrete performance, it was also considered to reduce the number of classes, but it leads to a minor number of samples to train the network.
So we find in SAR-Ship-Dataset [32] the best object detection dataset for satellite images with a significant number of samples for only one class.
The SAR Ship Dataset offers significant advantages over other types of satellite datasets, particularly for ship detection in complex environments using deep learning methods. First, it provides high-resolution imagery from Sentinel satellites, with multiple imaging modes and resolutions ranging from 3 to 25 meters. This diversity enables detailed ship detection across different environments. One of its key strengths is its ability to handle complex backgrounds such as harbors, islands, and buildings. By including such challenging scenarios, the dataset helps improve the robustness of ship detection models in real-world applications. The dataset also represents ships across multiple scales, accommodating variations in ship size and resolution, which enhances model training for detecting ships of different sizes, without the need of augmentation techinques. Finally, the dataset's volume, with 43,819 labeled ship chips, provides a broad and diverse training set, improving the generalization performance of detection models on new SAR images.

### 5.2.2 Model Architectures and Performance

Privacy preserving in object detection is a challenging task since we need to retrieve object from an image and it should be done in a "secure" way, where certain computations are performed on encrypted data to ensure privacy.

| Dataset | Focus | Number of Images | Images Size | Categories | Features |
|---|---|---|---|---|---|
| FAIR1M | Fine-grained object recognition in high-resolution remote sensing imagery. | 15000 | 1000-10000 | 5 categories with 37 sub-categories | Oriented Bounding Box, rich fine-grained category information, contains geographic information (latitude, longitude, resolution), and high image quality due to a careful data cleaning procedure. |
| DOTA-v2.0 | More common object detection | ∼12000 | 800-4000 | 15+ categories | Images collected from different sensors and platforms, wide variety of scales, orientations, and shapes, annotated by experts using arbitrary quadrilaterals, with Oriented Bounding Box |
| DIOR | object detection in optical remote sensing images, by detecting the solar radiation reflected from targets on the ground | ∼23000 | 800 | 20 | Large-scale in terms of categories, instance number, and total image number; large range of object size variations; big variations in imaging conditions, weathers, seasons, and image quality; high inter-class similarity and intra-class diversity. |
| xView | Object detection in overhead imagery about disaster relief missions | 1200 | 2000-4000 | 60 | Diverse collection of small, rare, fine-grained, and multi-type objects with bounding box annotation, also characterized by good geographic diversity |
| AI-TOD | The dataset was created to address the challenge of detecting tiny objects in aerial images | ∼28000 | 800 | 8 | The dataset is characterized by the small size of the objects, with a mean size of about 12.8 pixels, which is significantly smaller than other datasets. |

Table 5: Object Detection Dataset

Starting from the Faster-RCNN, we try to create a secure version of this architecture with a "smart" approach. We must consider the building of a Full Encrypted Faster-RCNN composed by a secure CNN, region proposal prediction on encrypted data, secure ROI pooling, and a secure detection network. The advantage of this design is that the entire object-detection task is offloaded to node who collects and process the image, ensuring data privacy but an high computational complexity and that RPN need the whole image to find bounding boxes and refine their coordinates. As a result, it can generate incorrect bounding boxes.
Another approach the encryption starts at the stage of ROI pooling, to obtain correct bounding boxes over the plain image. It can be done in 2 phases:

1. Split process on 2 steps: the initial feature extraction through the convolutional layers and the generation of region proposals via the RPN can be performed on plain data. This is because these operations do not directly reveal sensitive information about the objects in the image but rather process the image to identify areas of interest.
2. Processing exclusively on Encrypted Data: the object classification and bounding box regression stages involve more sensitive operations since they directly classify objects and determine their precise locations in the image. To ensure privacy preservation, these operations can be performed on encrypted data using HE. The classification and bounding box regression models need to be adapted to operate on encrypted data. This involves ensuring that the operations performed by these models are compatible with the HE scheme used.

While the first approach poses security risks due to the need to transmit plain data between systems, the

second approach is impractical because of the complexity involved in training and selecting optimal anchors. Specifically, step 4 *taking the top N locations and their objectness scores aka proposal layer* [3] relies on the argmax operation, which is not easily devolped in a HE environment [36] in a "black-box" framework like HElayers[1], as it requires calculating the IoU and identifying the index of the target bounding boxes with the highest overlap for each anchor box.
Different model architectures were considered, but a special focus is done to which obtain a good results while using few parameters. This is done in FHE lens, since FHE schemes add computational complexity and so to have a less time than possible from encryption of the network to the inference. A better solution is found in YOLO architectures which is exceptionally fast and simpler because it processes the entire image with a single forward pass through the network [23].
From original TinyissimoYOLO architecture [21], different version which predicts the centers of objects inside an image, with a decreasing number of parameters are developed which obtained the following results.

| | **Number of parameters** | **Memory Size (MB)** |
|---|---|---|
| TinyissimoFOMO (paper) | 945 392 | 5.67 |
| TinyissimoFOMOv1 | 159 088 | 3.05 |
| LeNetFOMO | 79 508 | 1.03 |

Table 6: YOLO architectures comparison

The network is designed to be extremely lightweight, with a low number of parameters and small memory consumption, making it suitable for deployment with limited resources.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
TinyissimoYOLO                           [1, 112]                  --
├─Sequential: 1-1                        [1, 16, 44, 44]           --
│    └─Conv2d: 2-1                       [1, 16, 88, 88]           160
│    └─LinearActivation: 2-2             [1, 16, 88, 88]           --
│    └─AvgPool2d: 2-3                    [1, 16, 44, 44]           --
├─Sequential: 1-2                        [1, 32, 22, 22]           --
│    └─Conv2d: 2-4                       [1, 32, 44, 44]           4,640
│    └─LinearActivation: 2-5             [1, 32, 44, 44]           --
│    └─AvgPool2d: 2-6                    [1, 32, 22, 22]           --
├─Sequential: 1-3                        [1, 64, 11, 11]           --
│    └─Conv2d: 2-7                       [1, 64, 22, 22]           18,496
│    └─LinearActivation: 2-8             [1, 64, 22, 22]           --
│    └─AvgPool2d: 2-9                    [1, 64, 11, 11]           --
├─Sequential: 1-4                        [1, 128, 5, 5]            --
│    └─Conv2d: 2-10                      [1, 128, 11, 11]          73,856
│    └─LinearActivation: 2-11            [1, 128, 11, 11]          --
│    └─AvgPool2d: 2-12                   [1, 128, 5, 5]            --
├─Sequential: 1-5                        [1, 112]                  --
│    └─Flatten: 2-13                     [1, 3200]                 --
│    └─Linear: 2-14                      [1, 256]                  819,456
│    └─LinearActivation: 2-15            [1, 256]                  --
│    └─Linear: 2-16                      [1, 112]                  28,784
==========================================================================================
Total params: 945,392
Trainable params: 945,392
Non-trainable params: 0
Total mult-adds (M): 28.96
==========================================================================================
Input size (MB): 0.03
Forward/backward pass size (MB): 1.86
Params size (MB): 3.78
Estimated Total Size (MB): 5.67
==========================================================================================
```

Figure 11: Summary of TinyissimoFOMO architecture

In this table 7 it can be seen the differences between original architecure and HE-compatible ones, with or

without bounding box. These experiments are done using a TinyissimoYOLO [21] architecture adapted to compute only the centers of objects inside an image, where the mAP consider a sample TP only if its centers is inside target bounding box.

| **Task** | **Activation** | **Pooling** | **mAP** |
|---|---|---|---|
| Bounding box | ReLu | MaxPooling | 0.3 |
| Bounding box | Square | AvgPooling | 0.005 |
| Bounding box | Linear | AvgPooling | 0.09 |
| Center | ReLu | MaxPooling | 0.67 |
| Center | Square | AvgPooling | 0.29 |
| Center | Linear | AvgPooling | 0.586 |

Table 7: YOLO and FOMO comparison w.r.t HE compatible operator

As we can see, simplifying the task clearly improves the network's performance. ReLU with MaxPooling consistently performs best in both bounding box and center prediction tasks, making it the preferred choice when considering mAP as the primary metric for accuracy. This combination seems to effectively balance feature extraction and pooling operations, preserving crucial information for both tasks. Square with AvgPooling shows poor performance, particularly for bounding box prediction. The significant drop in mAP suggests that this combination fails to capture and preserve the necessary features, making it unsuitable for these tasks in the context of HE compatible operations. Linear with AvgPooling performs moderately well, especially in the center prediction task. It could be a viable alternative in situations where ReLU might not be suitable, though its effectiveness varies depending on the specific task.
In Table 8, various model configurations are presented. These configurations were created by applying the square activation function and altering the dimensions of the kernel filters (specified by the first number), the stride (denoted by the number following '$s$'), and the padding (default is 1, otherwise indicated by the number following '$p$').

| **Network architecture** | **size (MB)** | **Number of parameters** | **mAP** |
|---|---|---|---|
| 5s1-3s1-3s1-3s1 | 5,63 | 946448 | 0,36 |
| 5s1-5s1-3s1-3s1 | 5,6 | 954604 | 0,355 |
| 5s1-5s1-5s1-5s1 | 4,07 | 594192 | 0,337 |
| 5s1-5s1-5s1 | 7,32 | 1400000 | 0,348 |
| 5s2-5s2-5s2 | 0,8 | 110000 | 0,33 |
| 5s2-5s1-5s1 | 1,42 | 241000 | 0,44 |
| 5s2-5s1-5s1p0 | 1,04 | 158000 | 0,46 |
| 7s2-5s1-5s1 | 1,03 | 161000 | 0,5 |

Table 8: FOMO with different layers configuration

The table illustrates that while increasing the complexity of a neural network (in terms of size and number of parameters) can improve performance, it is not always linear. Small adjustments to kernel size, stride, and padding can result in efficient models with better performance metrics, as demonstrated by the **7s2-5s1-5s1** configuration.
Other experiments are conducted by reducing the last layer of FC at the end of backbone, to reduce the number of parameters. Another way to reduce the last layer is to reduce the number of classes (C) and see how the network behave. The performance of a TinyissimoYOLO trained on DIOR data for 25 epochs are reflected on Figure 13.

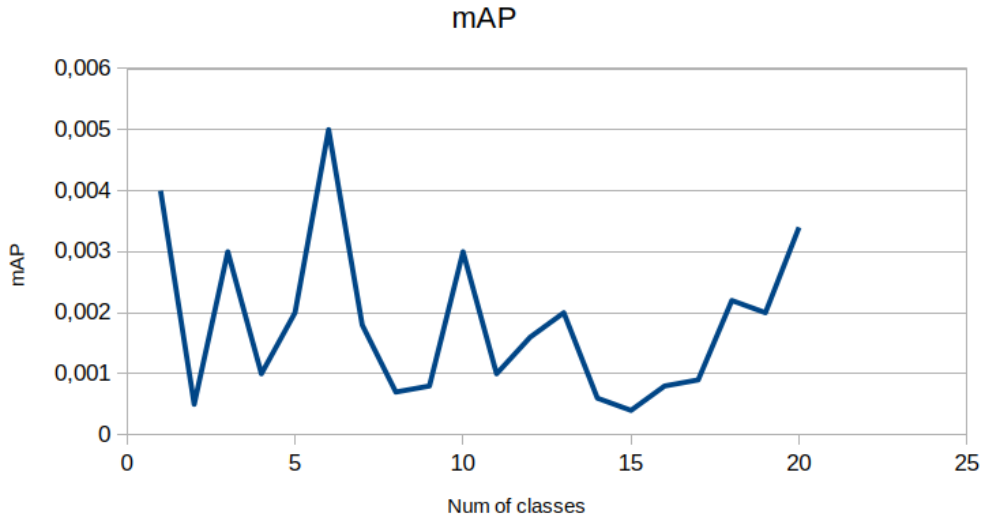



Figure 12: mAP w.r.t number of classes

The mAP values fluctuate as the number of classes increases, with no clear trend of improvement or decline. This suggests that reducing the number of classes doesn't consistently improve or worsen the model's precision. Reducing the number of classes leads to variability in precision. Although this strategy may lighten the network, the performance measured by mAP remains inconsistent, showing that fewer classes don't necessarily lead to better or worse model performance. This inconsistency might be due to the reduced number of training samples affecting the model's ability to generalize.

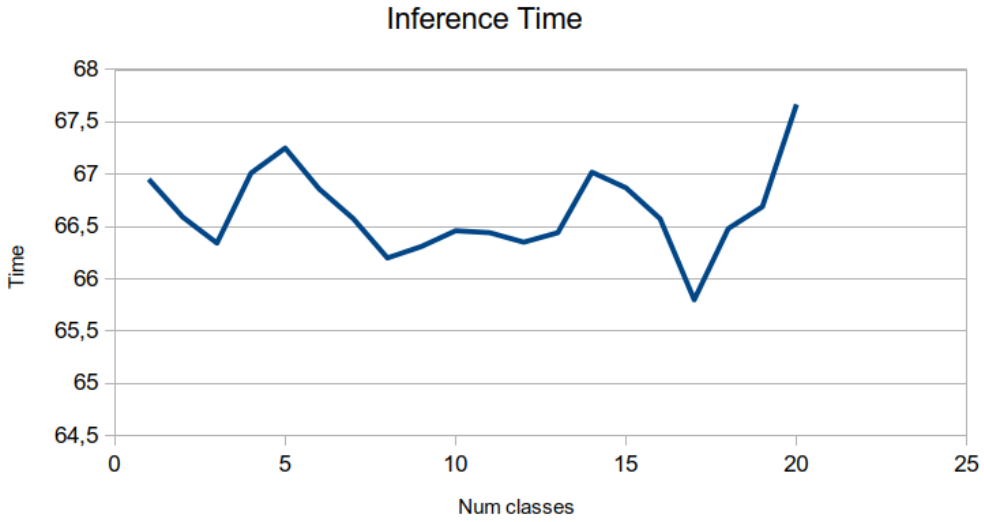



Figure 13: Inference time w.r.t number of classes

The inference time does not show a significant trend as the number of classes increases; it fluctuates slightly but remains relatively stable. The inference time remains relatively stable regardless of the number of classes. This indicates that while reducing classes may save on training time, it doesn't significantly affect the inference time. This stability is a positive aspect when considering a reduction in classes to create a lighter network, as it implies that the model's operational efficiency during inference is largely unaffected by the number of classes. Different experiments are done fine-tuning also NMS parameters and see how they influence the final mAP results. From these experiments it can be seen that also a simple network can reach good results, up to a mAP of 0.37.

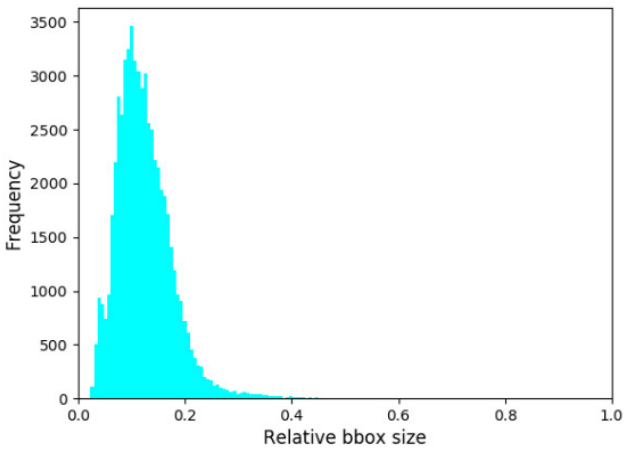
| Confidence threshold | Distance threshold | Num of predictions | mAP |
|---|---|---|---|
| 0 | 0 | 115570 | 0.35 |
| 0 | 0.2 | 99483 | 0.37 |
| 0 | 0.4 | 33918 | 0.37 |
| 0.2 | 0 | 12222 | 0.34 |
| 0.2 | 0.2 | 9222 | 0.35 |
| 0.2 | 0.4 | 8081 | 0.35 |
| 0.4 | 0 | 4780 | 0.23 |
| 0.4 | 0.2 | 4258 | 0.24 |
| 0.4 | 0.4 | 4078 | 0.23 |

Table 9: NMS fine-tuning performances

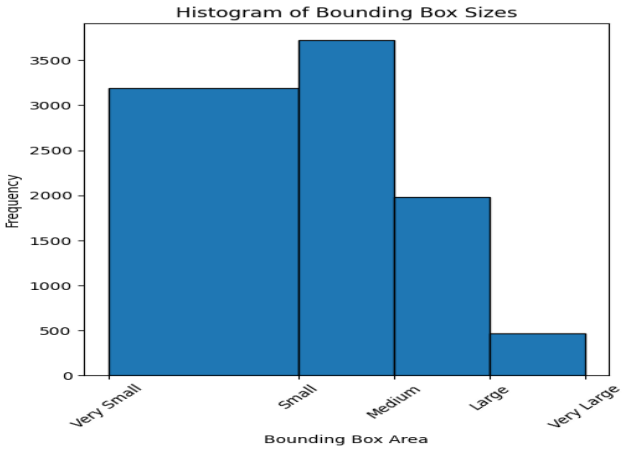
There is a clear trade-off between the number of predictions and the mAP. This is adjusted by NMS hyperparameters, prediction confidence and distances between predicted center of the same images. Higher confidence and distance thresholds reduce the number of predictions but can also lead to a loss in coverage, negatively impacting mAP. The best balance between the number of predictions and mAP appears to be with a low confidence threshold (0 or 0.2) and a moderate distance threshold (0.2). This setting retains a reasonable number of predictions while maintaining or slightly improving mAP. High confidence thresholds (e.g., 0.4) may lead to significant over-filtering, which reduces the number of predictions too much and decreases the mAP, suggesting that the model's ability to detect objects comprehensively is compromised.
Also reducing the number of prediction for each cell center (B) impact on performance. The mAP obtaines a drop of 10% dropping to the best model which obtain a mAP of 0.75 to a mAP of 0.66. When B is reduced, fewer bounding boxes are generated for each grid cell. This limits the model's ability to detect objects, especially in scenarios where there are multiple objects within the same grid cell or where objects have varied sizes and shapes. As a result, the model may miss certain objects or fail to predict their locations accurately because there are not enough predictions to account for all possible objects within the grid.
To evaluate the performance of different architectures, various tests are conducted because the new mAP metric alone doesn't provide information on how close the predictions are to the actual targets.
Additionally, an analysis is performed to measure how the predictions vary based on the size of the bounding boxes. This is important because smaller objects are generally harder to detect within larger images. The distance between the predicted and actual bounding boxes is calculated by considering the relative size of the bounding box compared to the overall image size.

$$\textit{Relative bboxes size} = \frac{\sqrt{w_{bbox} \cdot h_{bbox}}}{\sqrt{w_{img} \cdot h_{img}}}$$

The SAR-Ship dataset's bounding box sizes are distributed across four distinct categories, as shown in Figure 14. These categories are labeled as **Very Small**, **Small**, **Medium**, and **Large**. The classification into these groups is based on the relative size of the bounding boxes compared to the overall image size. Specifically, the bounding boxes are divided into intervals of 0.1, 0.15, 0.2, and 0.25 for each category. This categorization helps in understanding how objects of different sizes are represented within the dataset, which is crucial for evaluating the performance of detection models, particularly in identifying smaller objects that are often more challenging to detect.



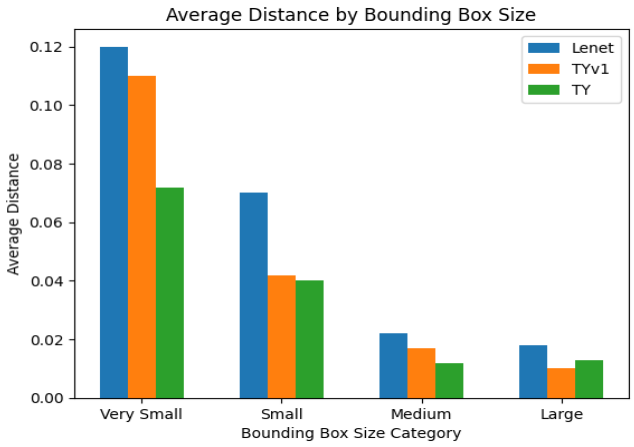

(a) Histogram of Relative Bounding Box Size



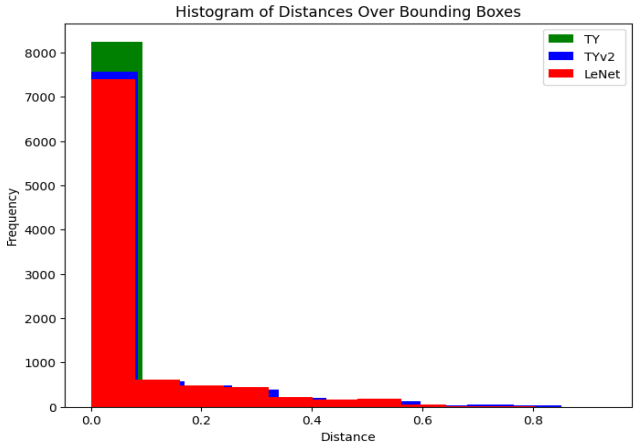

(b) Histogram of bounding box categories

Figure 14: Frequency of bounding box size per category

The network described above have the following performances:




(a) Average Distance by Bounding Box Size




(b) Histogram of Distances over Bounding Boxes

Figure 15: Predictions results of FOMO models

Figure 15a illustrates how different network architectures perform when detecting smaller objects. The results clearly indicate that heavier networks, which have more layers and parameters, are better equipped to detect small objects accurately. This is evidenced by the lower average distance between the predicted object center and the actual center, particularly when compared to lighter models. Heavier networks, consistently show a smaller average distance to the target, meaning that they can detect smaller objects with higher precision. This advantage is attributed to their more complex and deeper architectures, which allow for better feature extraction and pattern recognition between objects in different samples. As the size of the bounding boxes decreases, the distance between the predicted and actual object centers tends to increase when using lighter models. This suggests that lighter models struggle to maintain accuracy when tasked with detecting smaller objects, likely due to their reduced capacity to capture detailed features.
The majority of bounding box distances across all models are near 0 (Figure 15b), indicating that these models often produce predictions that closely match the ground truth bounding boxes. In particular, TinyissimoYOLO (green) shows a significantly higher frequency of very small distances with respect to other two models, suggesting that it tends to predict bounding boxes that are more accurate. As the distance increases, the frequency drops significantly for all models, but there is still a non-negligible number of instances where the predicted bounding boxes are further from the actual ones. In conclusion these graphs shows that the concentration of distances near zero suggests that these models can perform well also in environment where we have resource constraints due to high complexity introduced by HE.
These are the results by combining 3 and 5 models using ensemble approach described in section 4:

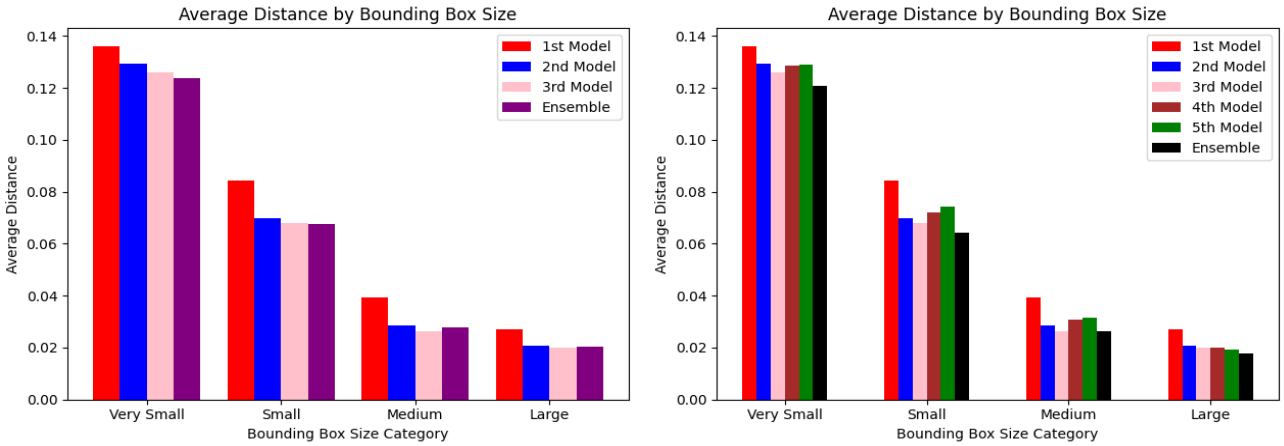



Figure 16: Average distance of ensembled models

Other experiments are conducted combining 5 models with different architectures with different number of parameters:

| Backbone Network | Description | Number of Parameters | mAP |
|---|---|---|---|
| Modello 1 | LeNet backbone on YOLO architecture | 79,508 | 0.34 |
| Modello 2 | Add convolution block and AvgPooling layer | 57,964 | 0.53 |
| Modello 3 | Different kernel size and number of filters | 106,656 | 0.6 |
| Modello 4 | Reducing the number of filters | 492,800 | 0.51 |
| Modello 5 | Increasing number of filters and different stride | 576,050 | 0.6 |

Table 10: Ensemble with FOMO architectures

From these graphs, it is clear how ensemble boost performances of smaller network introducing lower biased predictions, where by combining three or five models, the ensemble approach mitigates the weaknesses of individual models, particularly when detecting smaller objects. The ensemble models show a reduction in the average distance to the target compared to single models, suggesting improved robustness and accuracy.
Ensemble also gains benefits in terms of bounding box search. For this task, another experiment is conducted by combining 5 models with different architectures with a number of parameters ranging from 700,000 to 1,000,000.

| Backbone Network | Description | Number of Parameters | mAP |
|---|---|---|---|
| Modello 1 | TinyissimoYOLO | 961,840 | 0.59 |
| Modello 2 | Different number of filters | 714,608 | 0.6 |
| Modello 3 | Add convolutional block with AvgPooling layer | 1,007,408 | 0.57 |
| Modello 4 | Different kernel size and padding | 1,134,128 | 0.77 |
| Modello 5 | Add Dropout | 961,840 | 0.62 |

Table 11: Ensemble with different TinyissimoYOLO architectures

In order to evaluate performance, we relied on both the distance and the IoU between the predictions and the actual bounding boxes.

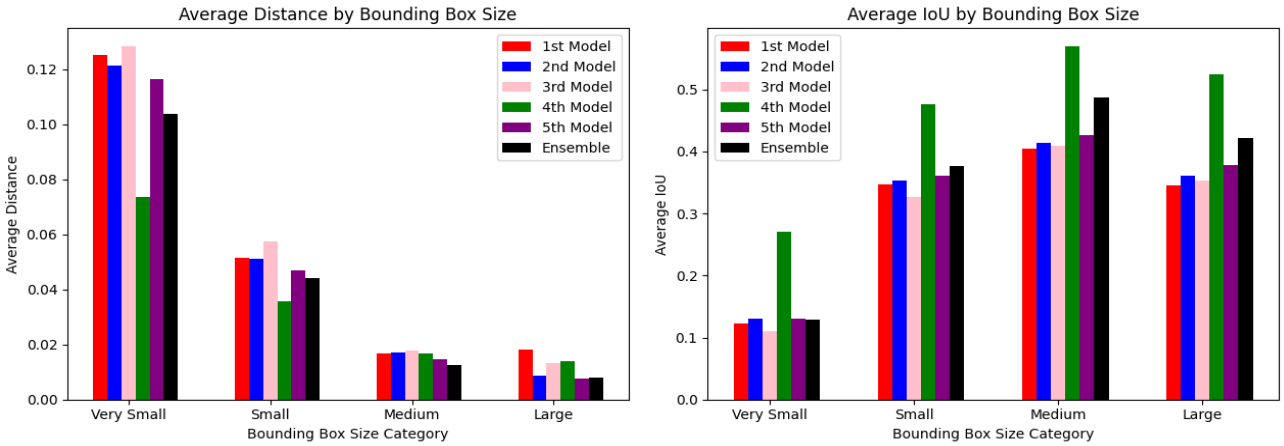



Figure 17: Ensemble with YOLO architectures

The results of the IoU analysis indicate that the 4th Model, the model with the highest number of parameters, consistently achieves the highest IoU across all bounding box size categories, demonstrating superior accuracy in predicting bounding boxes that closely match the ground truth. For Very Small objects, the 4th Model outperforms the others, suggesting its effectiveness in handling smaller, more challenging detections. In the Small and Medium categories, the 4th Model maintains its lead, with the Ensemble approach following closely behind, indicating that while the ensemble provides a balanced performance, individual models can still excel in specific scenarios. For Large bounding boxes, all models exhibit high IoU scores, with the ensemble slightly surpassing individual performances, highlighting the ensemble's ability to generalize well across varying object sizes. This suggests that while individual models may be optimized for specific tasks, the ensemble approach offers robust performance across different scales.
As discussed in previous section 4, bounding box task is a much more complex task and this is reflected by the fact that we need a much deepen architecture to understand correctly bounding box sizes. The ensemble approach outperforms individual YOLO models for large to medium-sized objects. It effectively reduces the average distance to the object center and increases the IoU, which demonstrates improved accuracy and precision. Despite the ensemble's overall performance, very small objects remain challenging, with higher average distances and lower IoU scores. This suggests that even with combined efforts, detecting and accurately predicting very small objects remains difficult.

### 5.2.3 Experiments under HE

In this section, we explore the application of HE in the context of neural network-based object detection tasks. Using the Pyhelayers library [1], we are able to apply HE schemes to neural networks in a "black box" manner. This approach significantly simplifies the integration of encryption, allowing the neural network to operate on encrypted data without requiring deep expertise in HE or cryptographic schemes. The Pyhelayers library handles the complexities of encryption and decryption, enabling a focus on model design and performance.
Experiments are done under a machine with 30GB of RAM, 200GB of swap and 16 cores. These experiments describe how HE influence inference time and computational complexity with respect to process data in plain.

| Model | Plain Inference Time | Encrypted Inference Time | RAM consumed |
|---|---|---|---|
| TinyYOLO | 459 ms | 2h 30m | 120 GB |
| TinyissimoFOMO | 74 ms | 12m | 30 GB |
| LeNetFOMO | 46 ms | 8 s | 8 GB |

Table 12: Complexity differences in a HE environment

This table remark how much optimization and speed improvement can decrease inference time under HE environment. Smaller architecture can reach good performances if task is make simpler and enhancing faster inference under HE environment.
The configuration of the HE context plays a crucial role in ensuring that encrypted data can be processed efficiently and securely during inference. Below is a detailed explanation of the specific HE parameters used for the object detection model implemented in this work:

| Parameter | Description | Value |
|---|---|---|
| Security Level | Defines the cryptographic strength, measured in bits | 128 bits |
| Integer Part Precision | Precision for representing integer values | 10 bits |
| Fractional Part Precision | Precision for representing fractional values | 38 bits |
| Number of Slots | Number of encrypted values that can be processed simultaneously | 8192 |
| Multiplication Depth | Max number of consecutive multiplications before precision degrades | 9 |
| Bootstrappable | Indicates whether bootstrapping is supported | False |
| Automatic Bootstrapping | Indicates if bootstrapping is done automatically | False |
| Rotation Keys Policy | Custom policy defining the required number of rotation keys | Custom, 17 keys |
| Rotation Offsets | Specific values for rotation offsets | [-2048, -1024, ..., 4096] |

Table 13: Summary of HE Parameters for the Model

The table outlines the critical parameters used in configuring the HE context for the object detection model. The security level of 128 bits is a standard setting that ensures strong cryptographic protection, providing sufficient defense against current computational threats. The integer part precision and fractional part precision values, set to 10 and 38 bits respectively, allow for accurate representation of both integer and fractional values, crucial for the precise operations required by neural networks during inference.
The number of slots is set to 8192, allowing the model to process a large batch of encrypted data simultaneously, which enhances the efficiency of operations in the encrypted domain. The multiplication depth of 9 ensures that multiple consecutive multiplications can be performed on the encrypted data without compromising decryption accuracy, making it suitable for the operations involved in object detection tasks.
Both bootstrappable and automatic bootstrapping are set to false, meaning that the model does not refresh the encrypted data after multiple operations, prioritizing computational efficiency over infinite-depth computations. Finally, the rotation keys policy defines a custom setup requiring 17 specific rotation keys, with the rotation offsets detailing the specific values needed to handle transformations on encrypted data efficiently during neural network operations.
This configuration strikes a balance between maintaining high-level security and enabling efficient encrypted inference, ensuring that the model can operate effectively without exposing sensitive data.
The experiments were designed to evaluate whether neural network performance remains consistent when operations are carried out in an encrypted domain. Our results indicate that the performance of the models before and after encryption is approximately the same, with only minor discrepancies. These mismatches are attributed to the prediction process being repeated twice over small convolutional neural networks (CNNs), which might introduce slight variations in the results. These are the statistics of the difference between plain predicions and decrypted predictions.

- Mean difference: $1.8419404144676623 \cdot 10^{-06}$
- Max difference: $2.0712371332276547 \cdot 10^{-05}$
- Min difference: $9.059018812873632 \cdot 10^{-10}$
- Std difference: $1.6256134035356975 \cdot 10^{-06}$

These results underscore the exceptionally high accuracy achieved in HE environments also with CKKS which does not have exact arithmetic, significantly surpassing other privacy-preserving techniques. The minimal differences observed in the calculations reflect the precision and reliability of HE in secure computations.

## 6. Conclusions

The results obtained confirm the feasibility of using neural networks based on HE for object detection tasks, demonstrating that despite the challenges introduced by HE, competitive performance can be achieved with appropriate modifications to traditional architectures. The introduction of linear functions or approximations and the balance between the number of parameters and the speed of inference have been crucial to adapt the models to the cryptographic context without compromising the accuracy of the predictions.
In particular, YOLO architectures with a reduced number of parameters have shown promise for simpler tasks, such as center prediction, demonstrating that less complex networks can still achieve good results. The ensemble approach also showed significant potential in improving the stability of predictions and reducing bias by combining the strengths of different models.
These results pave the way for further research and improvements in this field, suggesting that the integration of deep learning and homomorphic cryptography may become a viable solution for applications that require a high level of data security. In the future, the development of new optimization techniques and architectures specifically designed to operate in cryptographic environments could allow the capabilities and applications of these networks to be further extended.

### 6.1. Further work

The findings from the recent experiments suggest several promising directions for further research, particularly in the integration of HE within object detection networks. Currently, the process involves encrypting input data, performing computations in an encrypted domain, and then producing an encrypted output. This approach adheres to the principles of secure computation, but there is potential to refine and extend this methodology.
In object detection networks like YOLO, the input image is divided into a grid of cells, each responsible for predicting the presence of objects. YOLO, as described in the literature, allows each cell to produce multiple predictions (controlled by the parameter $B$), and selects the prediction with the highest confidence using a non-linear maximum function. However, under the HE framework, handling such non-linear operations presents certain challenges.
Two potential avenues for future work are:

- Maintaining a Single Prediction per Cell: Simplifying the network by limiting each cell to a single prediction could streamline the process within the HE environment, avoiding the complexities associated with non-linear functions.
- Approximating Non-Linear Functions with Polynomials: Another approach involves approximating non-linear functions, such as the maximum function, with polynomials, as suggested by existing mathematical research [36]. This could enable the network to operate fully within the HE framework without losing the ability to handle multiple predictions per cell.

My experiments thus far have involved decrypting the network's output and processing it post-decryption to aggregate predictions across the entire image. However, I discovered that this aggregation could potentially be conducted entirely within the HE environment, as it involves only linear operations. This opens up the possibility of performing the entire object detection process in a fully encrypted manner, maintaining security throughout.
Future work could focus on implementing and testing these approaches within a complete HE-based object detection pipeline. Additionally, further exploration into the capabilities of libraries like Helayers [1], particularly regarding operations on encrypted outputs, could lead to significant advancements in secure computation.

## 7. Bibliography and citations

## References


[1] Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, Hayim Shaul, and Omri Soceanu. Helayers: A tile tensors framework for large neural networks on encrypted data. *Proceedings on Privacy Enhancing Technologies*, 2023(1):325–342, January 2023.

[2] Ramy E. Ali, Jinhyun So, and A. Salman Avestimehr. On polynomial approximations for privacy-preserving and verifiable relu networks, 2024.

[3] Mark Campbell. Privacy-preserving computation: Doomed to succeed. *Computer*, 55:95–99, 08 2022.

[4] Imen Chakroun, Tom Vander Aa, Roel Wuyts, and Wilfried Verarcht. Privacy-preserving multi-party machine learning for object detection. In *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, pages 7–13, 2021.

[5] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 643–662, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[6] Soham De, Leonard Berrada, Jamie Hayes, Samuel L. Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale, 2022.

[7] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for detecting oriented objects in aerial images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[8] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[9] Alessandro Falcetta and Manuel Roveri. Privacy-preserving deep learning with homomorphic encryption: An introduction. *IEEE Computational Intelligence Magazine*, 17(3):14–23, 2022.

[10] Shruthi Gorantala, Rob Springer, and Bryant Gipson. Unlocking the potential of fully homomorphic encryption. *Commun. ACM*, 66(5):72–81, apr 2023.

[11] Rafik Hamza and Koji Zettsu. Investigation on privacy-preserving techniques for personal data. In *Proceedings of the 2021 ACM Workshop on Intelligent Cross-Data Analysis and Retrieval*, ICDAR '21, page 62–66, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. 08 2017.

[13] Chih-Hsien Hsia, Jen-Shiun Chiang, and Jing-Ming Guo. Multiple moving objects detection and tracking using discrete wavelet transform. In Hannu Olkkonen, editor, *Discrete Wavelet Transforms*, chapter 15. IntechOpen, Rijeka, 2011.

[14] Peter Ing. Introducing faster objects more objects aka fomo, 03/29/2022. Accessed: 08/29/24.

[15] Takumi Ishiyama, Takuya Suzuki, and Hayato Yamana. Highly accurate cnn inference using approximate activation functions over homomorphic encryption. *2020 IEEE International Conference on Big Data (Big Data)*, pages 3989–3995, 2020.

[16] Nathan Labbe. Detecting illegal gold mining sites in the amazon forest using deep learning to classify satellites images. *Degree Project in Computer Science, Second Cycle, 30 Credits*, 2021.

[17] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov, and Brendan McCord. xview: Objects in context in overhead imagery, 2018.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Jian Li and David Roy. A global analysis of sentinel-2a, sentinel-2b and landsat-8 data revisit intervals and implications for terrestrial monitoring. *Remote Sensing*, 9:902, 08 2017.

[20] Ke Li, Gang Wan, Gong Cheng, Liqiu Meng, and Junwei Han. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159:296–307, January 2020.

[21] Julian Moosmann, Marco Giordano, Christian Vogt, and Michele Magno. Tinyissimoyolo: A quantized, low-memory footprint, tinyml object detection network for low power microcontrollers. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, June 2023.

[22] Peter S. Nordholt, Nikolaj Volgushev, Prastudy Fauzi, Claudio Orlandi, Peter Scholl, Mark Simkin, Meilof Veeningen, Niek Bouman, and Berry Schoenmakers. D1.1 state of the art analysis of mpc techniques and frameworks. Technical report, Scalable Oblivious Data Analytics (SODA), 9 2017.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[24] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[26] Apple Machine Learning Research. Learning with privacy at scale. In *Proceedings of the 35th International Conference on Machine Learning*, ICML'22, pages 1344–1352, Austin, Texas, 2022. PMLR.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[28] Xian Sun, Peijin Wang, Zhiyuan Yan, Feng Xu, Ruiping Wang, Wenhui Diao, Jin Chen, Jihao Li, Yingchao Feng, Tao Xu, Martin Weinmann, Stefan Hinz, Cheng Wang, and Kun Fu. Fair1m: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery, 2021.

[29] Ganji Tejasree and Agilandeeswari Loganathan. An extensive review of hyperspectral image classification and prediction: techniques and challenges. *Multimedia Tools and Applications*, pages 1–98, 03 2024.

[30] Daliana Lobo Torres, Javier Noa Turnes, Pedro Juan Soto Vega, Raul Queiroz Feitosa, Daniel E. Silva, Jose Marcato Junior, and Claudio Almeida. Deforestation detection with fully convolutional networks in the amazon forest from landsat-8 and sentinel-2 images. *Remote Sensing*, 13(24), 2021.

[31] Jinwang Wang, Wen Yang, Haowen Guo, Ruixiang Zhang, and Gui-Song Xia. Tiny object detection in aerial images. pages 3791–3798, 2021.

[32] Yuanyuan Wang, Chao Wang, Hong Zhang, Yingbo Dong, and Sisi Wei. A sar dataset of ship detection for deep learning under complex backgrounds. *Remote Sensing*, 11(7), 2019.

[33] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[34] Tianying Xie, Hayato Yamana, and Tatsuya Mori. Che: Channel-wise homomorphic encryption for ciphertext inference in convolutional neural network. *IEEE Access*, 10:107446–107458, 2022.

[35] Ousmane Youme, Theophile Bayet, Jean Marie Dembele, and Christophe Cambier. Deep learning and remote sensing: Detection of dumping waste using uav. *Procedia Computer Science*, 185:361–369, 2021. Big Data, IoT, and AI for a Smarter Future.

[36] Peng Zhang, Ao Duan, and Hengrui Lu. An efficient homomorphic argmax approximation for privacy-preserving neural networks. *Cryptography*, 8(2), 2024.

[37] Wenting Zhao, Yunhong Wang, Xunxun Chen, Yuanyan Tang, and Qingjie Liu. Learning deep feature fusion for group images classification. In Jinfeng Yang, Qinghua Hu, Ming-Ming Cheng, Liang Wang, Qingshan Liu, Xiang Bai, and Deyu Meng, editors, *Computer Vision*, pages 566–576, Singapore, 2017. Springer Singapore.

[38] Yongjie Zhou, Jinbo Xiong, Renwan Bi, and Youliang Tian. Secure yolov3-spp: Edge-cooperative privacy-preserving object detection for connected autonomous vehicles. *2022 International Conference on Networking and Network Applications (NaNA)*, pages 82–89, 2022.

## Abstract in lingua italiana

In questa tesi esploriamo l'integrazione delle tecniche di conservazione della privacy nel dominio del Deep Learning, con un'attenzione specifica alle reti neurali applicate alla classificazione delle immagini e al rilevamento degli oggetti. Mentre le ricerche precedenti hanno affrontato ampiamente i metodi di conservazione della privacy nella classificazione delle immagini, il nostro lavoro si addentra nell'area meno esplorata del rilevamento degli oggetti. Ci concentriamo sullo studio e sull'adattamento di due importanti architetture, Faster-RCNN e YOLO, per lavorare con schemi di crittografia omomorfica (HE), in particolare CKKS, garantendo la massima privacy durante il calcolo.
I nostri contributi includono la progettazione e l'implementazione di adattamenti di questi modelli di rilevamento di oggetti che preservano la privacy, consentendo loro di eseguire l'inferenza e l'addestramento in modo sicuro con la crittografia omomorfa, salvaguardando così i dati sensibili. Inoltre, sottolineiamo l'ottimizzazione dell'architettura YOLO per ridurre l'overhead computazionale, una sfida critica dovuta alla natura intensiva delle operazioni HE. Grazie a queste ottimizzazioni, dimostriamo che è possibile ottenere un modello più efficiente e leggero, in grado di eseguire il rilevamento degli oggetti nel rispetto della privacy con tempi di inferenza e di addestramento ridotti, senza compromettere la sicurezza dei dati.
Questo lavoro non solo fa progredire il campo del deep learning rispettoso della privacy, ma fornisce anche indicazioni pratiche sulle sfide e sulle potenziali soluzioni per implementare sistemi di rilevamento degli oggetti sicuri ed efficienti utilizzando la crittografia omomorfa.

**Parole chiave:** Deep Learning a tutela della privacy, Immagini satellitari, Rilevamento di oggetti, Classificazione di immagini, Crittografia omomorfa

## Ringraziamenti

Vorrei esprimere la mia più profonda gratitudine a tutti coloro che mi hanno sostenuto durante questi incredibili anni accademici. Innanzitutto, sono profondamente grato alla mia famiglia, che non solo mi ha dato l'opportunità di proseguire gli studi in una città diversa, ma mi è stata accanto con grande sostegno ogni volta che ne ho avuto bisogno. Da mio padre, che mi ha insegnato l'importanza di lavorare sodo e come ogni piccolo sforzo ti porta sempre a creare qualcosa di più grande, a mia madre, che si è presa cura di me anche nei momenti più stressanti e mi ha sempre sostenuto in qualsiasi scelta facessi, a mia sorella, la mia spalla, che mi è sempre vicino ed è pronta sempre ad ascoltarmi, anche se a volte sono io che poi non ha voglia di raccontare. Un ringraziamento speciale va alla mia fidanzata Alessia, che mi ha fatto capire quanto una persona esterna possa entrare nella tua vita e migliorarla, in quest'ultimo anno mi ha spronato sempre di più e ha tirato fuori il meglio di me. Inoltre vorrei ringranziare i miei amici più cari Marco e Umberto e tutto il gruppo Losappio, amici che considero fratelli visto che ci conosciamo da quando eravamo bambini. Grazie per avermi fatto sempre strappare un sorriso, abbiamo condiviso tante avventure insieme e condividere con voi questo traguardo non ha prezzo.
Grazie a tutti miei amici della residenza, che mi hanno fatto sentire a casa già dal primo giorno, sopratutto vorrei ringranziare Gabri il mio coinquilino, il miglior coinquilino che potessi chiedere, grazie per tutte le risate che ci siamo fatti in questi anni e per il supporto che mi hai dato.
Infine, ringrazio di cuore tutti coloro che hanno contribuito a questa ricerca. Al mio professore e al suo assistente sono particolarmente grata per la loro preziosa guida, per i loro suggerimenti e per aver favorito il mio pensiero critico durante questo percorso. La vostra guida è stata fondamentale per dare forma a questo lavoro. Un ultimo pensiero va a mio nonno Salvatore, anche se ci siamo conosciuti poco, le tue storie mi hanno fatto capire quanto l'intraprendenza può portarti a fare grandi cose. Sei e sarai sempre il mio punto di riferimento.