Encrypt What Matters: ROI-Guided FHE for CNN Inference

Ali Backour[1], Juan Reyes[1], Jaime Punyed[1], and Ana Onoprishvili[1]

[1]Massachusetts Institute of Technology

Spring 2025

Abstract

Fully Homomorphic Encryption (FHE) enables privacy-preserving machine learning by allowing inference on encrypted data, bbut its high computational cost limits practical use. In this work, we introduce a novel approach to reduce this overhead by encrypting only the sensitive regions of input data, known as Regions of Interest (ROI), while keeping the rest in plain-text. Our approach leverages the local structure of Convolutional Neural Networks (CNN), where computations are spatially constrained, to ensure that encrypting a small sensitive region affects only a limited portion of the network. We provide a detailed theoretical analysis showing that ROI-based encryption can yield an order-of-magnitude speedup compared to full encryption. Our technique enables efficient and secure inference for applications like facial recognition, medical imaging, and document redaction, without requiring model retraining or compromising sensitive information.

1 Introduction

Machine learning has become a transformative force across a wide range of applications, from language processing to computer vision [1]. With the rapid advancement and widespread adoption of large language models (LLMs), an increasing number of users rely on these systems for tasks involving sensitive personal data, such as medical imaging, facial recognition, and document analysis [2, 3]. For instance, clinics may use Machine Learning models to better understand the statistics of some diseases, which requires processing patients' private data [4]. Similarly, security checkpoints often use face recognition models, which involve analyzing sensitive biometric data [5]. While running such models locally is infeasible due to hardware constrains, these data and computations are usually outsourced to remote servers which raises serious privacy concerns [6]. The servers are vulnerable to attacks that can compromise the user's privacy [7]. Some service providers may act under an honest-but-curious model, where data might be leaked.

To handle such risks, several research efforts focused on enabling Machine Learning models (Neural Networks) to run on encrypted data, ensuring that servers do not have access to sensitive data [8]. A powerful and relevant tool that was used for this purpose is Fully Holomorphic Encryption (FHE) which allows the computation on cipher texts without requiring decryption [9]. The idea is for the client to encrypt data before

feeding it to the LLM, let it evaluate the data homomorphically, and then decrypt the result on the client side. Although FHE was theoretically implemented in Gentry paper [9], its practical use for Machine Learning have faced several challenges. First of all, FHE schemes are inherently inefficient especially when applied to deep and complicated Neural Networks [8, 10]. Second, necessary Neural Networks components such as activation functions (e.g. ReLu, Sigmoid, Tanh ...) [11] are not polynomial operations and hence not supported in common FHE schemes. Third, FHE schemes often operate over integers, while neural networks typically require floating-point precision for weights and activations[12].

One of the first efforts to address these problems was CryptoNet [13] which demonstrated the feasibility of using FHE for Machine Learning Inference. In this paper, Dowlin et. al. encrypted the input images pixel by pixel, fed them to a 5-layer CNN, and then decrypted the output. The authors used polynomial activation functions (e.g. x^2) because it is FHE-friendly. They also implemented some software (using Chinese reminder theorem so that the model is more precise when encoding large numbers) and hardware (using SIMD operations) optimizations to make their model faster. Subsequent works, such as LOLA and FHE-DiNN [14, 15], introduced further optimizations and architecture adaptations to improve the performance of FHE. However, a major drawback of these approaches is the need to redesign the Neural Network architecture specifically to handle encrypted data, often requiring retraining complicated deep neural networks from scratch, a process that takes significant amount of time and resources.

More recently, Torus Fully Homomorphic encryption (TFHE) [16] introduced Programmable Bootstrapping, which allows arbitrary functions to be evaluated homomorphically using lookup tables. This scheme was later implemented by Zama AI [17], enabling encrypted inference without the need for architecture-specific retraining. However, they still suffer from a fundamental bottleneck: computational overhead . Even with hardware acceleration, FHE inference remains orders of magnitude slower than its plaintext equivalent, limiting its usability in time-sensitive scenarios[10, 8, 18].

In this work, we explore a complementary approach to reduce the performance overhead: partial encryption of input data. Specifically, we propose a method that encrypts only regions of interest (ROI) in the input, while leaving the remainder in plain text. This is particularly effective for convolutional neural networks (CNNs), where computation is inherently local and the presence of encrypted data affects only a limited portion of the circuit [19, 20]. We demonstrate that in CNNs, ROI-based encryption leads to significant reductions in circuit depth, gate count, and inference time, without retraining the model or compromising the privacy of sensitive regions. To the best of our knowledge, our approach, **Encrypt What Matters**, is the first to apply ROI-based selective encryption for CNN inference in the context of fully homomorphic encryption (FHE).

2 Preliminaries

2.1 Notation

Throughout this paper we will constantly be working over algebraic objects such as rings, fields and tori. We refer to elements of these objects using lower-case letters, e.g. $a \in \mathbb{Z}$.

We refer to vectors with bold lower-case letters, e.g. $\mathbf{a} \in \mathbb{Z}^n$. Lastly, we refer to the dot product between two vectors $\mathbf{a}, \mathbf{b}$ as $\langle \mathbf{a}, \mathbf{b} \rangle$, and the product between two members of a polynomial ring a, s as $a \cdot s$.

2.2 Learning With Errors Assumption (LWE)

Many of the FHE schemes that have been proposed so far rely on the learning with errors assumption (LWE) or natural extensions such as Ring-LWE (RLWE), Torus-LWE (TLWE) and General-LWE (GLWE). The purpose of this subsection is to introduce all these assumptions with some emphasis in TLWE and GLWE which are used in TFHE, the scheme that we use for our work. Regev [21] defined the LWE assumption as follows.

Learning With Errors (LWE). For a security parameter λ, let $n = n(\lambda)$ be an integer dimension, let $p = p(n) \leq \text{poly}(n)$ be a prime integer modulus, and $\chi : \mathbb{Z}_p \to \mathbb{R}^+$ be an error probability distribution on $\mathbb{Z}_p$. Now consider the following two distributions: for the first one, sample $(\mathbf{a}_i, b_i)$ uniformly over $\mathbb{Z}_p^{n+1}$; for the second one let $\mathbf{s} \in \mathbb{Z}_p^n$ be a secret vector uniformly sampled from $\mathbb{Z}_p^n$, then sample $\mathbf{a}_i$ uniformly from $\mathbb{Z}_p^n$ and sample e_i uniformly from χ, the resulting distribution is $(\mathbf{a}_i, \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i)$. The $\text{LWE}_{n,p,\chi}$ assumption is that it is infeasible to distinguish one distribution from the other.

It was under LWE that Gentry [9] showed the first ever construction of a fully homomorphic encryption scheme. Since then, there have been numerous propositions that aimed to reduce the time-space complexity. Many of these newer schemes strayed away from the traditional LWE definition above. For instance, the celebrated BGV [22] scheme uses a variant of LWE which is defined over rings. They define the Ring Learning with Errors assumption as follows.

Ring Learning With Errors Assumption (RLWE). For security parameter λ, let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2. Let $q = q(\lambda) \geq 2$ be an integer. Let $R = \mathbb{Z}[x]/(f(x))$ be a polynomial ring, and let $R_q = R/qR$. Let $\chi = \chi(\lambda)$ be a distribution over R. The $\text{RLWE}_{d,q,\chi}$ assumption is that it is infeasible to distinguish the following two distributions: In the first distribution, one samples (a_i, b_i) uniformly from R_q^2. In the second distribution, one first draws $s \leftarrow R_q$ uniformly, then samples $a_i \leftarrow R_q$ uniformly, $e_i \leftarrow \chi$ uniformly, and sets $b_i = a_i \cdot s + e_i$.

There are three important things to note here. The first one is that LWE and RLWE are syntactically the same, both assume that it is infeasible to distinguish between two distributions, one truly random and one that is not random but has some random noise. The second is that for the noisy distribution of LWE we are sampling n elements at random, while in RLWE we are just sampling 1 element at random. The last one is that $\mathbb{Z} = \mathbb{Z}[x]/(x^d + 1)$ for $d = 1$. From these similarities and differences the authors of BGV [22] define the General Learning with Errors problem.

General Learning With Errors Assumption (GLWE). For security parameter λ, let $n = n(\lambda)$ be an integer dimension, let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2. Let $q = q(\lambda) \geq 2$ be a prime integer. Let $R = \mathbb{Z}[x]/(f(x))$ be a polynomial ring, and let $R_q = R/qR$. Let $\chi = \chi(\lambda)$ be a distribution over R. The $\text{RLWE}_{d,q,\chi}$ assumption is that it is infeasible to distinguish the following two distributions: In the first distri-

bution, one samples $(\mathbf{a}_i, b_i)$ uniformly from R_q^{n+1}. In the second distribution, one first draws $\mathbf{s} \leftarrow R_q^n$ uniformly, then samples $\mathbf{a}_i \leftarrow R_q^n$ uniformly, $e_i \leftarrow \chi$ uniformly, and sets $b_i = \langle \mathbf{a}_i \cdot \mathbf{s} \rangle + e_i$.

The proposed FHE schemes are easier to implement under LWE but have worse performance than those under GLWE, suggesting a trade-off between performance/implementability and dimension/sample size, as pointed out in [22]. The generalization above can be further extended to work over R-mod ules, and its security has been proven via reductions in [23]. In our implementation we will use this R-module version since we will be working with the TFHE scheme [16], a GSW-based FHE which works over the real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ which is a module of the integer ring.

2.3 Programmable Bootstrapping

The construction of a FHE scheme remained an open problem for more than 30 years until Gentry's scheme [9] was devised. One of the groundbreaking ideas behind the construction was bootstrapping. In short, GLWE-based schemes have the property that every operation will increase the range of the noise distribution χ until there is a point where the GLWE cipher-texts become too noisy to maintain correctness. To circumvent this problem Gentry proposed the idea of homomorphically decrypting data that has been re-encrypted using a different key, effectively reducing the noise while keeping information secure. This technique called bootstrapping is present in all of the GLWE-based schemes, transforming them from leveled homomorphic encryption schemes into FHE [24].

The magic (and usefulness) of TFHE comes from what the authors call *Programmable Bootstrapping*. This technique allows the scheme to evaluate arbitrary functions implemented as a look-up table during the bootstrapping phase of the scheme. For clarity and completeness we show a digest of their procedure ahead.

2.3.1 Blind Rotations

Let the plaintext $\mu = \mu_0 + \mu_1 x + \cdots + \mu_{n-1} x^{n-1}$ be such that each of the coefficients have $\log(q)$ bits and are of the form shown in Figure 1 below.

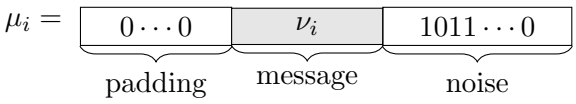

Figure 1: Plaintext format

Notice that the same message can be represented by several different plaintexts with varying amounts of noise. Let $\bar{\nu}_j$ be the noiseless representation corresponding to message encoding j, i.e. the message section of $\mu_i = j$.

The idea behind blind rotations is to use these $\bar{\nu}_j$ to define what the authors in [17] call the *static polynomial*. Let $v = \bar{\nu}_0 + \cdots + \bar{\nu}_{q-1} x^{q-1}$ be the static polynomial, then multiplying v by the monomial $x^{-\mu_i}$ will shift the position of the coefficient of x^{μ_i} to

be the independent coefficient, i.e. ν_i is now the independent coefficient. All we have to do now is extract the coefficient and continue with a noise-reduced encryption of μ_i. Naturally, these multiplications of polynomials are performed homomorphically as part of the bootstrapping procedure to satisfy the security of the scheme.

2.3.2 Look-up Table Evaluation

Note that when we are performing the blind rotation, we are obtaining the noise-reduced encryption of μ_i, because we decided this was the desired output when defining the static polynomial. Essentially, we are mapping each possible encryption to their noise-reduced version and looking it up using the static polynomial as a look-up table. We can abuse this idea to evaluate arbitrary functions while performing bootstrapping (hence the name *programable bootstrapping*).

Formally, consider the function Upper defined in [17] which maps i to $\bar{\nu}_i$. Let $f : \mathscr{D} \to \mathscr{F}$ be an arbitrary function from domain $\mathscr{D}$ to image $\mathscr{F}$, and consider the encoding and decoding functions of both $\mathscr{D}$ and $\mathscr{F}$ with respect to $\mathbb{Z}/q\mathbb{Z}$ which is where the coefficients of the plaintext lie. So,

$$\begin{aligned} \text{Encode} &: \mathbb{Z}/q\mathbb{Z} \to \mathscr{D} & \text{Decode} &: \mathscr{D} \to \mathbb{Z}/q\mathbb{Z} \\ \text{Encode'} &: \mathbb{Z}/q\mathbb{Z} \to \mathscr{F} & \text{Decode'} &: \mathscr{F} \to \mathbb{Z}/q\mathbb{Z}. \end{aligned}$$

Then, let T be a function such that $T = \text{Encode'} \circ f \circ \text{Decode} \circ \text{Upper}$, and consider the new static polynomial $v = T[0] + \cdots + T[q-1]x^{q-1}$. Naturally, the result of performing the blind rotation procedure with this new static polynomial is the evaluation of the noise-reduced plaintext on f. Therefore, we can use this static polynomial as a look-up table for any arbitrary function f provided the encoding and decoding functions. For visualization refer to the diagram below [17].

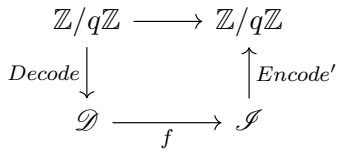

Figure 2: Arbitrary function implementation

3 Our Approach

In this work, we propose a method to reduce the computational overhead of homomorphic inference by selectively encrypting only the sensitive portions of the input, also known as Regions of Interest (ROI). The key insight is that only certain parts of the input actually require privacy protection. For instance, in an image of a person, the background often carries little privacy risk compared to the face.

Let's assume we have a model $\mathcal{M}$, which we model as circuit $\mathcal{C}$, takes n inputs and

that our data is $\mathbf{x} = (x_1, x_2 \ldots, x_n)$. Let the output of the model be $y = \mathcal{C}(\mathbf{x})$. Traditionally, one will homomorphically encrypt $\mathbf{x}$ to get the ciphertext $\mathbf{c} = (c_1, c_2 \ldots c_n)$ then we decrypt $\mathcal{C}(\mathbf{c})$ to get y as shown in Figure 3 below [13].

Figure 3: Traditional Protocol to use FHE in Machine Learning inference

In contrast, let's assume that we only consider $\tilde{\mathbf{x}} = (x_1, x_2 \ldots x_m)$ to be sensitive while the rest of the data $\mathbf{x}' = (x_{m+1}, x_{m+2}, \ldots x_n)$ to be the plaintext (background). Let's partially evaluate $\mathcal{C}$ on $\mathbf{x}'$ to get $\mathcal{C}'$ which takes m input. Now, we repeat the traditional protocol of homomorphically encrypting $\tilde{\mathbf{x}}$ to get ciphertext $\tilde{\mathbf{c}}$ and decrypting $\mathcal{C}'(\tilde{\mathbf{c}})$, as shown in Figure 4 below.

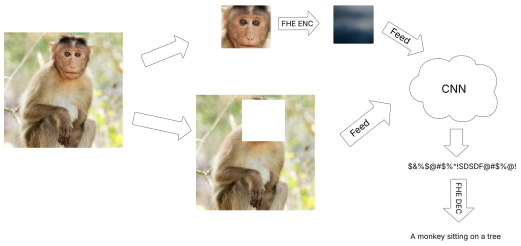

Figure 4: ROI protocol to use FHE in Machine Learning inference

As a result, our approach will gain a significant speed-up if size($\mathcal{C}'$) < size($\mathcal{C}$). However, this is not always the case. If $\mathcal{M}$ is a Fully-connected Neural Network with several hidden layers [25], the resulting circuit $\mathcal{C}'$ will have almost the same size as $\mathcal{C}$. For instance, consider the neural network in Figure 5 below. Notice that only one bit of the input is encrypted but $\mathcal{C}'$ will be exactly the same as $\mathcal{C}$ except for the first layer. We refer to this phenomenon as *Propagation of Sensitivity* which measures the extent to which encrypted input values influence downstream computations throughout the network. Formally, we define the *Propagation of Sensitivity* at layer i as the number of gates whose computation depends on any encrypted input, denoted by $|S_i|$. The total propagation across a depth-d network is then given by:

$$\text{Total Propagation} = \sum_{i=1}^{d} |S_i|$$

which quantifies how much of the circuit is influenced by encrypted regions throughout the network.

In architectures like Fully-connected Neural Network, this propagation is maximal (i.e Total Propagation $\approx$ Size of the Network).

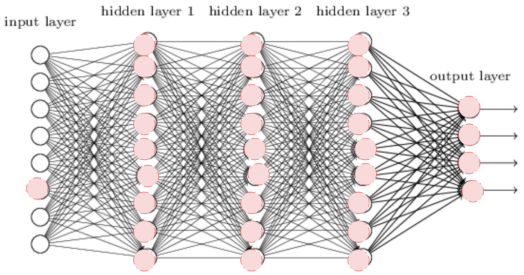

Figure 5: Fully connected neural network with one input encrypted

Fortunately, by its nature, CNN will minimize the propagation of sensitivity [19, 20]. Let's assume that our input $\mathbf{x}$ is an $n \times n$ image and let $\tilde{x}$ to be a square part of that image of size $m \times m$, and that the kernel of the CNN is $k \times k$. Hence the circuit $\mathcal{C}$ representing $\mathcal{M}$ will take n^2 input. Notice that each input is connected to at most $2k \times 2k$ gates in the following layers. Moreover, each consecutive m^2 input gates are connected to at most $(m+2k)^2$ gates in the next layer. Hence, the propagation per layer is equal to $(m+2k)^2$. As a result, the circuit $\mathcal{C}'$ will take m^2 inputs, and the second layer will be of size $(m+2k)^2$. More specifically, the d^{th} layer will be of size $(m+2kd)$, growing linearly in d. Thus, for models with more compatible parameters (such as a lower depth), the propagation is diminished more, obtaining better results. Figre 6 below demonstrates the propagation of sensitivity per layer for an image in a CNN

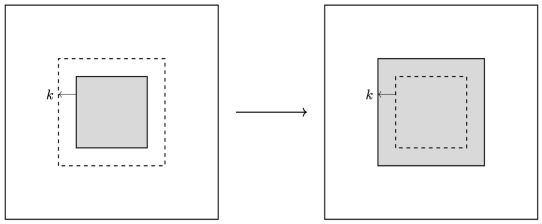

Figure 6: Propagation of encrypted data from layer i to $i+1$ in $\mathcal{C}$. The shaded square represents the encrypted pixels

4 Theoretical Speed up

In this section, we will discuss the theoretical speed up that our scheme will bring. Again, let the input be an $n \times n$ image, for which we only want to encrypt the sensitive data S, which we assume is an $m \times m$ square. Let the CNN have d layers and a filter of size $k \times k$.

First, there is a straightforward speedup for the Client that wants to send this encrypted message, as they do not need to encrypt the whole $n \times n$ image, but instead a much smaller $m \times m$ square.

More importantly, on the server side, the CNN circuit $\mathcal{C}$ should be partially evaluated to get $\mathcal{C}'$ which will be significantly smaller, as we will demonstrate below. Both $\mathcal{C}$ and

$\mathcal{C}'$ have addition and multiplication gates (scaling) between encrypted and clear data, as well as activation function gates applied to both encrypted and non-encrypted inputs, which, in the encrypted case, are evaluated via programmable bootstrapping using lookup tables which is significantly slower.

As discussed in Section 3, the fan-out of each encrypted gate (corresponding to a pixel) in the i^{th} layer extends to neighboring gates in the $(i+1)^{th}$ layers due to the local connectivity of convolutional operations. More specifically, if S_i is the set of gates in the i^{th} layer in $\mathcal{C}$ that have S as one of their ancestors, then as shown in section 3, $|S_{i+1}| \leq (m'+2k) \times (m' \times 2k)$ where $|S| = m \times m$. Hence, $|S_i| \leq (m+2ki) \times (m+2ki)$.

Recall that in $\mathcal{C}$, each layer will have $k^2 \cdot n^2$ scaling gates (i.e. multiplication between encrypted data and clear data (weights)). However, in $\mathcal{C}'$, the $(i+1)^{th}$ layer will only have $k^2(m+2ki)^2$ scaling gates.

For the addition gates, each application of a convolutional filter requires summing the results of multiple scaling gates. These addition gates operate over encrypted data only when the corresponding multiplications involve inputs within the encrypted region, which grows to a size of $(m+2ki) \times (m+2ki)$ at the i^{th} layer. Each addition is followed by an activation function (e.g., ReLU as shown in Figure 1), which is the most computationally expensive operation when applied to encrypted data due to the need for programmable bootstrapping [16, 17]. Since there is one activation per addition, we incur $(m+2k(i+1))^2$ encrypted activation function evaluations per layer, each requiring a lookup table evaluation as described in Section 2.4.2.. Hence we have:

$$\text{number of scaling gates:} \quad \sum_{i=0}^{d-1} k(m+2ki)^2 \leq kd(m+2kd)^2$$

$$\text{number of activation gates:} \quad \sum_{i=0}^{d-1} (m+2k(i+1))^2 \leq d(m+2kd)^2$$

As a result, we get the following comparison table:

Gate Type	Normal FHE	Our Implementation
Activation Functions on Encrypted	dn^2	$d(m+2kd)^2$
Multiplication Clear and Encrypted	kdn^2	$kd(m+2kd)^2$
Addition Clear and Encrypted	dn^2	$d(m+2kd)^2$

Since the bottleneck is found in the activation functions over encrypted data, we expect to get a speedup of around $\frac{(m+2kd)^2}{n^2}$. For more explicit results, for a filter of size 3x3, a Region of Interest of size 256x256, a CNN of depth 10, and an original image of size 1024x1024, we obtain the speedup of $\frac{(256+2\cdot 3\cdot 10)^2}{1024^2} \sim 1/9$

5 Deployability

Our algorithm was designed with practical deployment in mind, and we developed a full implementation plan tailored to the *Concrete* framework [26], one of the most promising

FHE libraries currently available. Our core idea, combining an encrypted $m \times m$ region of interest with a plaintext $n \times n$ background, fits naturally within Concrete's hybrid encryption model, where function parameters can be individually marked as encrypted or clear-text.

However, current limitations in the *Concrete* framework prevent complete realization of our design. Specifically, the framework lacks support for dynamic matrix manipulation and non-square matrix multiplication, either of which would enable inserting encrypted regions into a larger plain-text input [26]. While our conceptual implementation is sound, these constraints restrict our ability to run a full end-to-end evaluation at this time. These limitations highlight areas where future research can be done to unlock broader classes of privacy-preserving algorithms.

Below, we include a pseudo-code sketch of our approach. It demonstrates how an encrypted ROI can be efficiently embedded within a plain-text image using only matrix multiplications, operations that are friendly to FHE circuit design:

Algorithm 1 Insert Encrypted Matrix Using Matrix Multiplication

Require: model, $x_{\text{clear}} \in \mathbb{R}^{n \times n}$, $x_{\text{enc}} \in \mathbb{R}^{m \times m}$, insertion point (h, w)

Initialize zero matrices $P \in \mathbb{R}^{n \times m}$, $Q \in \mathbb{R}^{n \times m}$
for $i = 0$ to $m - 1$ **do**
    $P[h + i, i] \leftarrow 1$
end for
for $j = 0$ to m-1 **do**
    $Q[w + j, j] \leftarrow 1$
end for
$x_{\text{inserted}} \leftarrow P \cdot x_{\text{enc}} \cdot Q^{\top}$
$x_{\text{out}} \leftarrow x_{\text{clear}} + x_{\text{inserted}}$
return model(x_{out})

In addition, Concrete's compiler-based function evaluation opens the door to partial execution strategies. More specifically, we can compile the model forward function f into a circuit C first, then evaluate it partially on x_{clear} in plain-text to obtain a residual circuit C'. After that, we perform homomorphic evaluation of C' on x_{enc}.

While *Concrete* does not yet support circuit partial evaluation [26], we believe that this work provides a clear use case and motivation for adding support for partial circuit evaluation and dynamic tensor multiplication in FHE libraries like *Concrete*.

A separate practical challenge involves identifying which regions of the input should be considered sensitive. One straightforward approach is to let the user to manually blur areas they want to be protect. A more robust and automated solution is to run a lightweight model locally to detect identifiable information (e.g. faces, license plates, etc.) [27, 28].

6 Application

Our scheme offers a more efficient solution to protecting private user data while using powerful machine learning models by enabling efficient encrypted inference on sensitive

regions of input data. We highlight several promising applications:

6.1 Private Facial Recognition

A common privacy concern in facial recognition systems is the exposure of biometric data to third-party services [5]. With our selective FHE approach, a user can encrypt only the region of an image containing a face while keeping the rest of the image in plain text.

6.2 Medical Imaging and Diagnosis

In medical settings, some parts of an image (e.g., a CT scan or an MRI) may reveal sensitive patient information [4] while other regions may not. Our system enables clinicians to share encrypted scans with remote diagnostic models that can operate on clear data and homomorphically evaluate only the protected regions.

6.3 Surveillance Footage Processing

In surveillance scenarios, it is often necessary to identify people or license plates without exposing their identities broadly [5]. Selective encryption allows systems to encrypt only these sensitive areas.

6.4 Autonomous Driving

Autonomous vehicles often need to offload computation to cloud services while respecting privacy [29]. Our approach allows vehicles to encrypt only regions of interest (e.g., pedestrians or faces) and send the rest of the scene in plain text.

6.5 Document Redaction and Smart OCR

Documents may contain a mix of sensitive and non-sensitive information. For example, a scanned legal document might include a person's SSN alongside public case details. By encrypting only the sensitive segments, our system can homomorphically extract or classify redacted data while allowing the rest of the document to be processed normally.

7 Conclusion

In this work, we introduced Encrypt What Matters, a novel method to reduce the computational cost of fully homomorphic encryption (FHE) inference by selectively encrypting only the sensitive regions of input data, a strategy we refer to as Region-of-Interest (ROI) encryption. Our key insight lies in recognizing that in many machine learning applications, such as facial recognition, medical imaging, and document analysis, where only parts of the input actually require encryption to preserve privacy. By exploiting the inherent locality of convolutional neural networks (CNNs), our approach substantially reduces circuit size, gate count, and inference latency without requiring any modification or retraining of the underlying model.

We formally analyzed the theoretical benefits of ROI encryption and showed that, under reasonable assumptions on kernel size and network depth, our method achieves a multiplicative speedup proportional to the ratio between encrypted and un-encrypted

regions. This reduction is especially impactful in settings where only a small portion of the input contains sensitive information, as demonstrated in our analysis.

Despite its conceptual promise, our attempt to implement this approach using the Concrete framework was limited by current library constraints, especially the lack of support for non-square matrix operations and partial circuit evaluation. Nonetheless, our proposed algorithm is compatible with the design principles of existing FHE compilers and can be readily integrated once these limitations are addressed.

As FHE continues to evolve toward greater usability and efficiency, we believe ROI encryption offers a practical middle ground: preserving strong privacy guarantees where necessary, while enabling tractable deployment of machine learning models in real-world, latency-sensitive settings. Future work will focus on extending this approach to more general architectures and exploring automated mechanisms for identifying sensitive regions in unstructured data.

8 Who did what

Ali: Came up with the original idea and attempted several times to implement the code using concrete but did not end up working. In addition, he wrote the Approach section, the Deployability section, as well as parts of the introduction.

Juan: Surveyed existing bibliography. Understood and share with the teammates the theory behind the scheme that was being used for the implementation, as well as pointed out the bottlenecks and areas that could be improved. Additionally wrote the Preliminaries section and part of the introduction.

Jimmy: Calculated the theoretical speedup and wrote its section in the paper. Also worked on attempting to implement the code by trying to obtain the circuit of the CNN and change it accordingly.

Ana: Found open source implementations of programmable bootstrapping and tried some of them but only concrete had a working Python environment. She wrote the application part of this paper and helped with the introduction.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023. Accessed: 2025-05-13.

[3] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[4] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in Bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[5] C. Garvie, A. Bedoya, and J. Frankle, "The perpetual line-up: Unregulated police face recognition in america," tech. rep., Georgetown Law Center on Privacy & Technology, October 2016. Accessed: 2025-05-13.

[6] J. Xu, Z. Li, W. Chen, Q. Wang, X. Gao, Q. Cai, and Z. Ling, "On-device language models: A comprehensive review," *arXiv preprint arXiv:2409.00088*, 2024. Accessed: 2025-05-13.

[7] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, (New York, NY, USA), pp. 199–212, Association for Computing Machinery, 2009.

[8] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "ngraph-he: a graph compiler for deep learning on homomorphically encrypted data," in *Proceedings of the 16th ACM international conference on computing frontiers*, pp. 3–13, 2019.

[9] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC '09: Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169–178, 2009.

[10] N. Neda, A. Ebel, B. Reynwar, and B. Reagen, "Ciflow: Dataflow analysis and optimization of key switching for homomorphic encryption," *arXiv preprint arXiv:2311.01598*, 2023. Accessed: 2025-05-13.

[11] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *arXiv preprint arXiv:2109.14545*, 2022. Accessed: 2025-05-13.

[12] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in cryptology–ASIACRYPT 2017: 23rd international conference on the theory and applications of cryptology and information security, Hong kong, China, December 3-7, 2017, proceedings, part i 23*, pp. 409–437, Springer, 2017.

[13] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of The 33rd International Conference on Machine Learning*, pp. 201–210, 2016.

[14] A. Brutzkus, A. Elisha, Y. Goldberg, and R. El-Yaniv, "Low latency privacy preserving inference," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 812–821, 2019.

[15] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Advances in Cryptology – ASIACRYPT 2018* (D. Wang and M. Yung, eds.), vol. 11274 of *Lecture Notes in Computer Science*, pp. 483–512, Springer, 2018.

[16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus." Cryptology ePrint Archive, Paper 2018/421, 2018.

[17] I. Chillotti, M. Joye, and P. Paillier, "Programmable bootstrapping enables efficient homomorphic inference of deep neural networks." Cryptology ePrint Archive, Paper 2021/091, 2021.

[18] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster cryptonets: Leveraging sparsity for real-world encrypted inference," *arXiv preprint arXiv:1811.09953*, 2018. Accessed: 2025-05-13.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012. Accessed: 2025-05-13.

[20] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems 29 (NeurIPS 2016)*, pp. 4898–4906, Curran Associates, Inc., 2016. Accessed: 2025-05-13.

[21] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, 2005.

[22] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping." Cryptology ePrint Archive, Paper 2011/277, 2011.

[23] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.

[24] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications." Cryptology ePrint Archive, Paper 2022/1602, 2022.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] Zama, "Concrete: A rust-based fully homomorphic encryption framework." https://github.com/zama-ai/concrete, 2023. Documentation available at https://docs.zama.ai/concrete. Accessed: 2025-05-13.

[27] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. Accessed: 2025-05-13.

[28] T. Orekondy, M. Fritz, and B. Schiele, "Connecting pixels to privacy and utility: Automatic redaction of private information in images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8466–8475, IEEE, 2018. Accessed: 2025-05-13.

[29] R. Hamon, H. Junklewitz, J. I. Sanchez Martin, D. Fernandez Llorca, E. Gomez Gutierrez, A. Herrera Alcantara, and A. Kriston, "Artificial intelligence in automated driving: An analysis of safety and cybersecurity challenges," Tech. Rep. JRC127189, European Commission, Joint Research Centre, 2022. Accessed: 2025-05-13.