

Encrypt What Matters: ROI-Guided FHE for CNN Inference

Ali Backour¹, Juan Reyes¹, Jaime Punyed¹, and Ana Onoprishvili¹

¹Massachusetts Institute of Technology

Spring 2025

Abstract

Fully Homomorphic Encryption (FHE) enables privacy-preserving machine learning by enabling inference on encrypted data, but its high computational cost limits practical deployment. 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 Motivation

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 certain 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 constraints, these 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 exposed.

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 used for this purpose is Fully Homomorphic Encryption (FHE), a class of encryption schemes that allows computations

(mainly addition and multiplication) to be performed directly on encrypted data, without ever decrypting it [9]. The idea is for the client to encrypt the data before feeding it to the model, have the model evaluate it homomorphically, and send the encrypted result back to the client, who can decrypt it.

Although FHE was theoretically implemented in Gentry paper [9], its practical use for machine learning has faced several challenges. First of all, FHE schemes are inherently inefficient, especially when applied to deep 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 natively 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].

2 Related Work

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 Remainder 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 deep neural networks from scratch, a process that takes a significant amount of time and resources.

More recently, Torus Fully Homomorphic encryption (TFHE) [16] introduced Programmable Bootstrapping, which allows arbitrary functions (e.g., neural networks activation 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, based on Zama AI scheme [19], 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 [20, 21]. We demonstrate that in CNNs, ROI-based encryption leads to significant reductions in circuit depth (computational graph), 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 neural network inference in the context of fully homomorphic encryption (FHE).

3 Preliminaries

In this section, we briefly review the fundamentals of Fully Homomorphic Encryption (FHE) and the computational model used throughout this work. Our goal is to establish the notation and concepts needed to understand how convolutional neural networks (CNNs) can be evaluated on encrypted data, and how partial encryption of inputs affects the size of the resulting arithmetic circuit.

3.1 Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is a class of public-key encryption schemes that allows arbitrary computations to be performed directly over encrypted data. Formally, an FHE scheme consists of the following algorithms:

- **KeyGen**: generates a public-secret key pair (pk, sk) .
- **Enc**: encrypts a plaintext m under pk to produce a ciphertext $c = Enc_{pk}(m)$.
- **Eval**: given a function f (consists of addition and multiplication) and ciphertexts $(c_1, \dots, c_n)$, produces a ciphertext

$$c_f = Eval_{pk}(f(c_1, \dots, c_n))$$

such that c_f encrypts $f(m_1, \dots, m_n)$.

- **Dec**: decrypts c_f under sk , yielding $Dec_{sk}(c_f) = f(m_1, \dots, m_n)$.

To evaluate a function under FHE, the function must be expressed as an arithmetic circuit composed of addition and multiplication gates. The main bottleneck in FHE is multiplicative depth and the need for *bootstrapping*, a noise-refreshing operation that restores ciphertext integrity at significant computational cost.

3.2 TFHE and Programmable Bootstrapping

Our work builds on the TFHE family of schemes, particularly the Concrete framework by Zama AI[19], which provides fast implementations of programmable bootstrapping (the procedure that refreshes a ciphertext when its noise grows too large during homomorphic evaluation). In TFHE:

- additions and multiplications by plaintext values (scaling) are inexpensive
- ciphertext-ciphertext multiplications are significantly more expensive,
- nonlinear functions (e.g., ReLU) are evaluated through *programmable bootstrapping*, which applies an arbitrary lookup table (LUT) while refreshing noise, which is also expensive.

Thus, the dominant cost in encrypted neural network inference arises from ciphertext-ciphertext multiplications and activation-function bootstraps.

3.3 Arithmetic Circuit Model for neural networks

We represent a neural network $\mathcal{M}$ as an arithmetic circuit consisting of linear operations (addition and scaling gates) and nonlinear activation gates (e.g. ReLU). Let $\mathcal{C}_{\mathcal{M}}$ (homomorphic circuit) be the circuit generated by Concrete framework [19] from $\mathcal{M}$ which has multiply-by-plaintext (scaling) gates as well as programmable bootstrapping gates (LUTs) for nonlinear activations. For simplicity, we assume both $\mathcal{M}$ and $\mathcal{C}_{\mathcal{M}}$ takes $\mathbf{x} = (x_1, \dots, x_n)$ as input (e.g. pixel of images) and let $y = \mathcal{M}(\mathbf{x})$

4 Our Approach

In this work, we propose *Encrypt What Matters*, 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.

Using our notation, traditionally one will encrypt $\mathbf{x}$ to get the ciphertext $\mathbf{c} = (c_1, \dots, c_n) = \text{Enc}(x)$ using an FHE scheme, feed it to the homomorphic model circuit $\mathcal{C}_{\mathcal{M}}$ then decrypt $\mathcal{C}_{\mathcal{M}}(\mathbf{c})$ to get y , as shown in Figure 1 below [13]. The correctness of this follows directly from the guarantees of the FHE scheme $\text{Dec}(\mathbf{c}) = \mathcal{C}_{\mathcal{M}}(\mathbf{x}) = y$

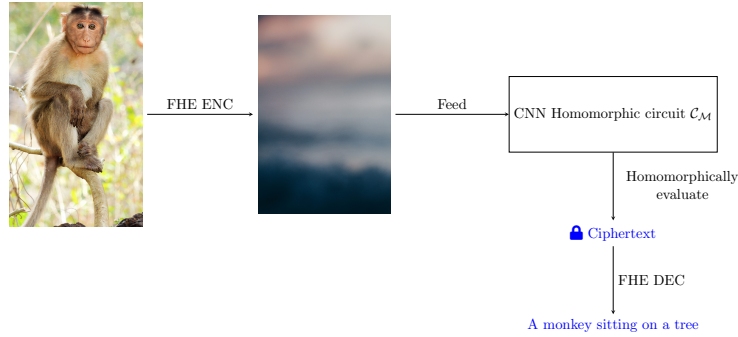


Figure 1: Traditional Protocol to use FHE in machine learning inference.

In contrast, we propose to merely encrypt one part of the input while treating the other parts as plaintext. More specifically, let us assume that we only consider $\tilde{\mathbf{x}} = (x_1, \dots, x_m)$ to be sensitive while the rest of the data $\mathbf{x}' = (x_{m+1}, \dots, x_n)$ is not. As before, we have the model $\mathcal{M}$, we will partially evaluate $\mathcal{M}$ on $\mathbf{x}'$ to get $\mathcal{M}'$, which is modeled as an arithmetic circuit (similar to $\mathcal{M}$) that takes m inputs. Now, we repeat the traditional protocol on $\mathcal{M}'$ and $\tilde{\mathbf{x}}$. More specifically, we encrypt $\tilde{\mathbf{x}}$ using FHE to get ciphertext $\tilde{\mathbf{c}} = \text{Enc}(\tilde{\mathbf{x}})$ then we evaluate and decrypt $\mathcal{C}_{\mathcal{M}'}(\tilde{\mathbf{c}})$ as shown in Figure 2 below.

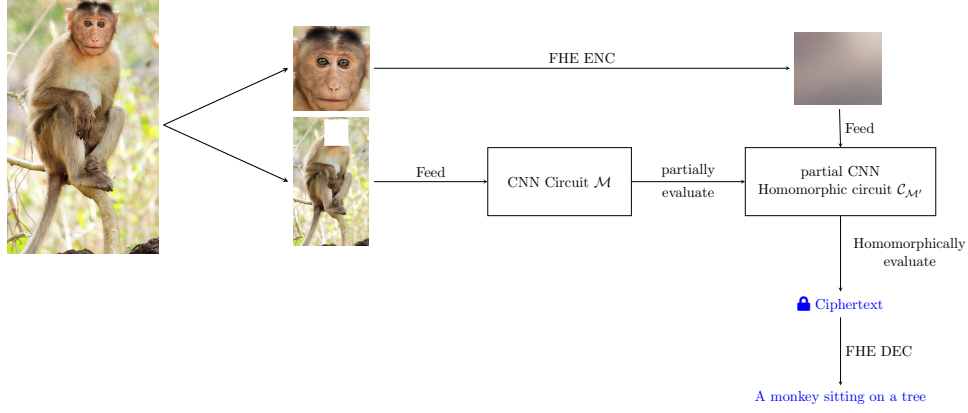


Figure 2: ROI protocol to use FHE in machine learning inference

First, we will show the correctness of our approach. Notice that $\mathcal{M}(\mathbf{x}) = \mathcal{M}'(\tilde{x})$ by definition of partial evaluation. On the other hand, we have

$$Dec(\tilde{\mathbf{c}}) = \mathcal{C}_{\mathcal{M}'}(\tilde{x}) = \mathcal{C}_{\mathcal{M}}(\mathbf{x}) = y$$

Notice that the partial evaluation is very efficient since it is run on normal neural network circuit before doing any encryptions. As a result, given that the size of $\mathcal{M}'$ is significantly smaller than that of $\mathcal{M}$, we get a significant speedup since the bottleneck (evaluating $\mathcal{C}_{\mathcal{M}}$ on $\mathbf{c}$) is now being done over $\mathcal{C}_{\mathcal{M}'}$, which is much smaller and hence faster.

However, this does not necessarily hold for all architectures. For instance, if $\mathcal{M}$ is a fully connected neural network with several hidden layers [22], the resulting circuit $\mathcal{M}'$ will have almost the same size as $\mathcal{M}$ (as shown in Figure 3). Notice that only one bit of the input is encrypted, but $\mathcal{M}'$ will be the same as $\mathcal{M}$ except for the first layer.

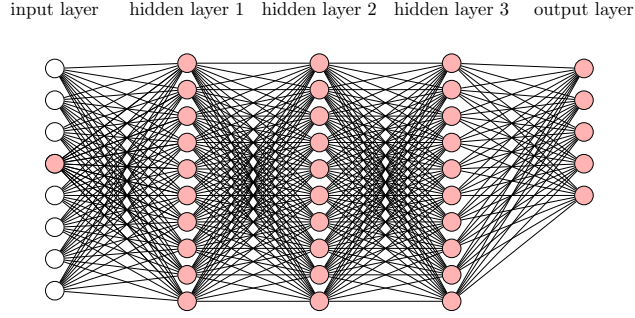


Figure 3: Fully connected neural network with single encrypted input (in red). The red nodes represent the $\mathcal{M}'$

To formally measure this, we define the *Propagation of Sensitivity*, which measures the extent to which encrypted input influences downstream computations throughout the network.

For an input $x = (x_1, \dots, x_n)$, and a model $\mathcal{M}$, define the *Propagation of Sensitivity* of a model $\mathcal{M}$ with respect to input x_i as

$$PS_i(\mathcal{M}) = \frac{|\mathcal{M}'_i|}{|\mathcal{M}|}$$

where $\mathcal{M}'_i \subseteq \mathcal{M}$ is the circuit formed by the minimal set of gates in $\mathcal{M}$ that's reachable from x_i (in other words, the size of $\mathcal{M}'$ is the size of the circuit $\mathcal{M}$ partially evaluated at $x_{j \neq i} = 1$). We define the overall propagation of sensitivity as:

$$PS(\mathcal{M}) = \mathbb{E}[PS_i(\mathcal{M})]$$

One can observe that

$$\text{speedup} \propto \frac{1}{PS(\mathcal{M})}$$

For example, one can show $PS(\mathcal{M}_{\text{FNN}})$ is close to 1 (as shown in Figure 3 above) and hence our method will not yield a significant speedup.

Fortunately, by its nature, CNN minimizes the propagation of sensitivity [20, 21] as we will show below. Approximately, we have $PS(\mathcal{M}_{\text{CNN}}) = \mathcal{O}(\frac{(2kd)^2}{n})$ where k is the filter size and d is the depth of the network (usually both are very small compared to n). More concretely, let us assume that our input $\mathbf{x}$ is an $n \times n$ image and let $\tilde{x}$ be a square part of that image of size $m \times m$, and that the kernel of the CNN is $k \times k$. 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, as shown in Figure 4 below. Hence, the propagation per layer is equal to $(m + 2k)^2$. As a result, the second layer of $\mathcal{M}'$ will be of size $(m + 2k)^2$. More specifically, the d^{th} layer will be of size $(m + 2kd)^2$, growing linearly in d . Thus, for models with more compatible parameters (such as a lower depth), the propagation is diminished more, obtaining better results.

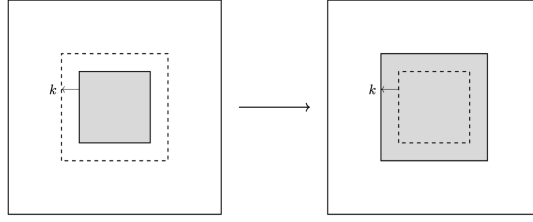


Figure 4: Propagation of encrypted data from layer i to $i + 1$ in $\mathcal{M}$

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.) [23, 24].

5 Theoretical Analysis

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 $k \times k$ (for simplicity we omit the pooling and the fully-connected layers).

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 $\mathcal{M}$ should be partially evaluated to get $\mathcal{M}'$ which will be significantly smaller, as we will demonstrate below. Both $\mathcal{C}_{\mathcal{M}}$ and $\mathcal{C}_{\mathcal{M}'}$ 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, $|S_{i+1}| \leq (m' + 2k) \times (m' + 2k)$ 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 $\mathcal{C}_{\mathcal{M}'}$, 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:

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

As a result, we get the following comparison table:

Gate Type	Normal FHE	Our Implementation
Activation Functions on ciphertext	dn^2	$d(m + 2kd)^2$
Multiplication (ciphertext $\times$ plaintext)	kdn^2	$kd(m + 2kd)^2$
Addition (ciphertext $\times$ plaintext)	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 $\frac{n^2}{(m+2kd)^2}$. 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 $\frac{1024^2}{(256+2 \cdot 3 \cdot 10)^2} \approx 9$.

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

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 like Transformers and pruned neural networks, as well as exploring automated mechanisms for identifying sensitive regions in unstructured data.

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] 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.
- [20] 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.
- [21] 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.

- [22] 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.
- [23] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018. Accessed: 2025-05-13.
- [24] 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.