information

*Article*

# A Homomorphic Encryption Framework for Privacy-Preserving Spiking Neural Networks

**Farzad Nikfam [1,*], Raffaele Casaburi [1], Alberto Marchisio [2], Maurizio Martina [1] and Muhammad Shafique [2]**

[1] Department of Electrical, Electronics and Telecommunication Engineering, Politecnico di Torino, 10129 Torino, Italy; s266053@studenti.polito.it (R.C.); maurizio.martina@polito.it (M.M.)
[2] eBrain Lab, Division of Engineering, New York University, Abu Dhabi P.O. Box 129188, United Arab Emirates; alberto.marchisio@nyu.edu (A.M.); muhammad.shafique@nyu.edu (M.S.)
* Correspondence: farzad.nikfam@polito.it

**Abstract:** Machine learning (ML) is widely used today, especially through deep neural networks (DNNs); however, increasing computational load and resource requirements have led to cloud-based solutions. To address this problem, a new generation of networks has emerged called spiking neural networks (SNNs), which mimic the behavior of the human brain to improve efficiency and reduce energy consumption. These networks often process large amounts of sensitive information, such as confidential data, and thus privacy issues arise. Homomorphic encryption (HE) offers a solution, allowing calculations to be performed on encrypted data without decrypting them. This research compares traditional DNNs and SNNs using the Brakerski/Fan-Vercauteren (BFV) encryption scheme. The LeNet-5 and AlexNet models, widely-used convolutional architectures, are used for both DNN and SNN models based on their respective architectures, and the networks are trained and compared using the FashionMNIST dataset. The results show that SNNs using HE achieve up to 40% higher accuracy than DNNs for low values of the plaintext modulus $t$, although their execution time is longer due to their time-coding nature with multiple time steps.

**Keywords:** deep neural network (DNN); spiking neural network (SNN); homomorphic encryption (HE); Brakerski/Fan-Vercauteren (BFV); Norse; Pyfhel; privacy preserving; FashionMNIST; Python; PyTorch; privacy; security; safety; machine learning; artificial intelligence



**Citation:** Nikfam, F.; Casaburi, R.; Marchisio, A.; Martina, M.; Shafique, M. A Homomorphic Encryption Framework for Privacy-Preserving Spiking Neural Networks. *Information* **2023**, *14*, 537. https://doi.org/10.3390/info14100537

Academic Editor: Panayiotis Kotzanikolaou

Received: 2 August 2023
Revised: 28 September 2023
Accepted: 29 September 2023
Published: 1 October 2023



## 1. Introduction

Machine learning (ML) has witnessed significant development in recent years, finding diverse applications in various sectors such as robotics, automotive, smart industries, economics, medicine, and security [1–3]. Several models based on the structure of the human brain have been implemented [4], including the widely used deep neural networks (DNNs) [5,6] and spiking neural networks (SNNs) [7], which emulate the functioning of neurons relatively better than DNNs [8]. These models require large amounts of data to be trained and reach high accuracy. However, if such data are collected from users' private information, such as personal images, interests, web searches, and clinical records, the DNN deployment toolchain will access sensitive information that could be mishandled [9]. Moreover, the large computational load and resource requirements for training DNNs have led to outsourcing the computations on the cloud, where untrusted agents may undermine the algorithms' confidentiality and intellectual property of the service provider. Note that encrypting the data transmission in the communication from client to server using common techniques such as advanced encryption standard (AES) would not solve the issues, because untrusted agents on the server side have full access to the sensitive data and DNN model. Among privacy-preserving methods, homomorphic encryption (HE) employs polynomial encryption to encrypt input data, perform computations, and decrypt the output. Because the computations are conducted in the encrypted (ciphertext) domain, the ML

algorithm and data remain confidential as long as the decryption key is unknown to the adversary agents. However, common HE-based methods focus on traditional DNNs, and studying the impact and potential of encryption techniques for SNNs is still unexplored.

In this work, we deploy the Brakerski/Fan-Vercauteren (BFV) HE scheme [10] for SNNs, and compare with its application to DNN architectures [11]. From the experimental results, we observed that the SNN models working on encrypted data yield better results than traditional DNN models, despite the increased computational time due to the intrinsic latency of SNNs that simulate human neurons.

Our novel contributions are summarized as follows (see an overview in Figure 1):

- We design an encryption framework based on the BVF HE scheme that can execute privacy-preserving DNNs and SNNs (Section 3).
- The encryption parameters are properly selected to obtain good tradeoffs between security and computational efficiency (Section 3.4).
- We implement the encryption framework, evaluate the accuracy of encrypted models, and compare the results between DNNs and SNNs. We observe that the SNNs achieve up to 40% higher accuracy than DNNs for low values of the plaintext modulus $t$ (Section 4).

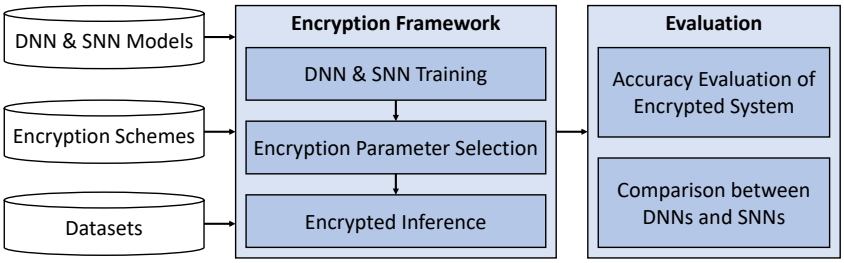


**Figure 1.** Overview of our novel contributions.

Paper organization: Section 2 contains the background information of the methods and algorithms used in this work, which are DNNs, SNNs, and HE, with a particular focus on the BFV scheme. Section 3 discusses the proposed encryption framework for DNNs and SNNs and describes our methodology for selecting the encryption parameters. Section 4 reports the experimental results and a discussion on the comparison between DNNs and SNNs when using HE. Section 5 concludes the paper.

## 2. Background

### *2.1. Deep Neural Networks and Convolutional Neural Networks*

DNNs, whose functionality is shown in Figure 2a, are a class of artificial neural networks composed of multiple layers of interconnected nodes called neurons. These networks are designed to mimic the structure and functioning of the human brain. DNNs are characterized by depth, referring to the many hidden layers between the input and output. This depth allows DNNs to learn complex patterns and representations from data, enabling them to solve intricate problems in fields such as image and speech recognition, natural language processing, and more.

Convolutional neural networks (CNNs) [12] are a specialized type of DNN designed to efficiently process grid-like data, such as images or time series. CNNs apply filters to input data, capturing local patterns and features. This allows CNNs to extract hierarchical representations from visual data, enabling object detection, image classification, and image generation tasks. CNNs have revolutionized the field of computer vision and have been widely adopted in various applications, including autonomous driving, medical imaging, and facial recognition.

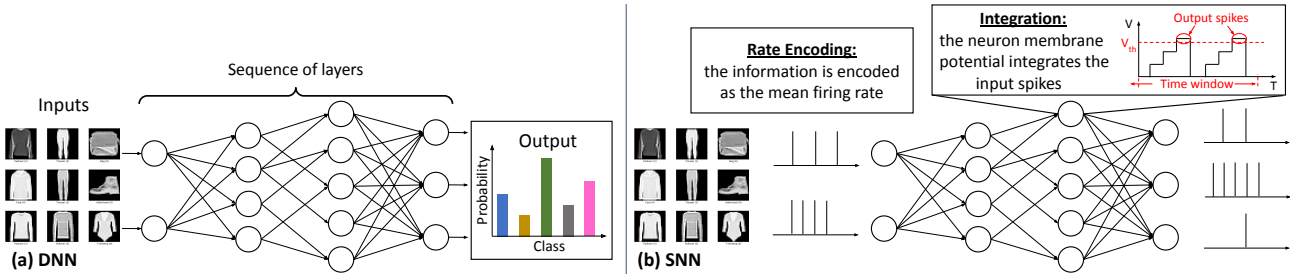


**Figure 2.** Overview of (**a**) the functionality of a DNN and (**b**) the functionality of an SNN.

### *2.2. Spiking Neural Networks*

SNNs [13–15] are a type of neural network model that aim to replicate the behavior of biological neurons. Unlike traditional DNNs that use continuous activation values, SNNs communicate through discrete electrical impulses called spikes. As shown in Figure 2b, these spikes encode the timing and intensity of neuron activations, allowing for more precise and efficient information processing [16–19]. SNNs are particularly suited for modeling dynamic and time-varying data, as they can capture the temporal aspects of input signals. This enables SNNs to excel in temporal pattern recognition, event-based processing, and real-time sensory processing [20–23]. SNNs provide an efficient and brain-inspired computing paradigm for executing ML workloads. However, processing SNNs on traditional (Von Neumann) architectures demands high energy consumption and execution time. To overcome these issues, designers have developed specialized hardware platforms such as neuromorphic chips to execute SNNs in a fast and efficient manner. Compared to non-spiking DNNs, the communication between neurons in SNNs is discrete through spike trains, whereas DNNs have continuous activation values. The key advantage of SNNs is that computations are executed only in the presence of spikes. If the spikes are sparse in time, SNNs can save a large amount of energy compared to the non-spiking DNNs that process continuous values. By emulating the spiking behavior of biological neurons, SNNs offer a promising avenue for understanding and replicating the computational capabilities of the human brain. Because conventional ML datasets typically lack any form of temporal encoding, an additional encoding step is necessary to introduce the required temporal dimension [24]. In the case of SNNs, input spikes are treated as a sequence of tensors consisting of binary values [25–27].

### *2.3. Homomorphic Encryption and Brakerski/Fan-Vercauteren Scheme*

HE is a cryptographic technique that allows computations on encrypted data without decryption [28,29]. A popular scheme used in HE is the BFV scheme [10] (see Figure 3). This scheme leverages polynomial encoding to enable encrypted data manipulation. In this scheme, the client encrypts their sensitive input data using a public key provided by the server [30,31]. The server computes the encrypted data using specialized algorithms that maintain the encryption. The encrypted results are then returned to the client, who can decrypt them using their private key to obtain the desired outputs. The BFV scheme supports addition and multiplication operations on encrypted variables, preserving the algebraic structures necessary for computation. By employing this scheme, sensitive data remain protected throughout the computation process, ensuring privacy and security [32–35]. HE comes in different variants, such as partially HE (PHE), somewhat HE (SHE), and fully HE (FHE), each offering different levels of computation capabilities on encrypted data [10,36–39].

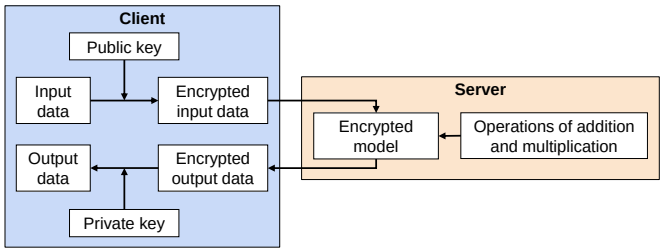


**Figure 3.** A fully homomorphic encryption (FHE) scheme.

The BFV scheme is a type of FHE, which means that operations are fully encrypted, both on multiplications and additions. Consequently, there is no possibility of obtaining intermediate information during the process. To explain this concept more clearly, we can look at an example using an equation. In this case, we will apply homomorphic invariance only to addition, but FHE applies the same logic to multiplication as well. Our basic equation is Equation (1). Let us assume it undergoes a homomorphic transformation (encryption) represented as Equation (2). Let us calculate the result by choosing random values for $x$ and $y$ (see Equation (3)). Calculating both sides of Equation (1), we obtain Equations (4) and (5). Applying the homomorphic transformation of Equation (2), we obtain Equations (6) and (7). We obtained the same result on both sides of the equation, despite the homomorphic transformation applied in the middle. This is what HE accomplishes. In the case of the BFV scheme and FHE in general, homomorphism applies to both additions and multiplications.

$$f(x+3y) = f(x) + f(3y) \tag{1}$$

$$f(z) = 5z \tag{2}$$

$$\begin{cases} x = 2 \\ y = -6 \end{cases} \tag{3}$$

$$f(2+3\cdot(-6)) = f(2) + f(3\cdot(-6)) \tag{4}$$

$$f(-16) = f(2) + f(-18) \tag{5}$$

$$-80 = 10 - 90 \tag{6}$$

$$-80 = -80 \tag{7}$$

## 3. Proposed Encryption Framework

In this work, (see Figure 4), we implement a LeNet-5 CNN [11] and its equivalent SNN variant. For the dataset, we leveraged FashionMNIST [40] (see Figure 5), which is similar to MNIST [41] but consists of 10 classes of clothing items (note that we adopt the same test conditions as widely used by the SNN research community where the typical evaluation settings [42] use the spiking LeNet and datasets such as MNIST and Fashion MNIST). The hardware system used for conducting the experiments consisted of a Tesla P100-PCIE GPU, an Intel(R) Xeon(R) Gold 6134 CPU @ 3.20GHz, and 100 GB of RAM. We developed the code in Python, utilizing the PyTorch framework [43], the Pyfhel library for the encryption [44], and the Norse library to implement the SNN [45].

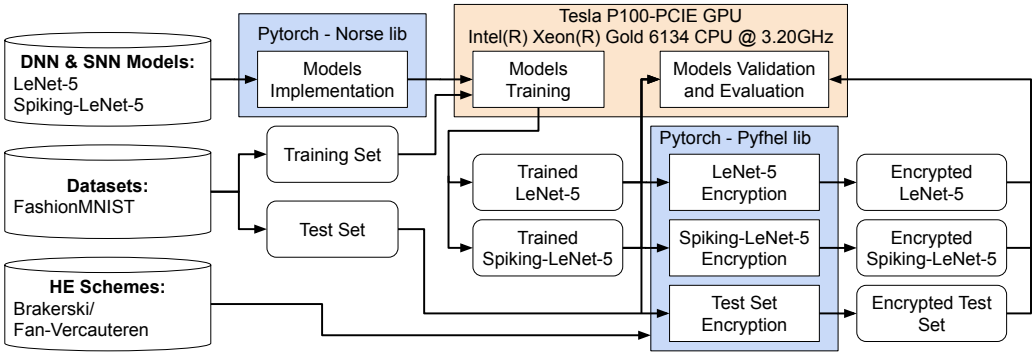


**Figure 4.** Our proposed encryption framework with the experimental setup.

### *3.1. FashionMNIST*

FashionMNIST [40] (see Figure 5) is a widely used dataset in computer vision and machine learning. It serves as a benchmark for image classification tasks and is a variation of the classic MNIST dataset. Instead of handwritten digits, FashionMNIST consists of grayscale images of various clothing items, such as shirts, dresses, shoes, and bags. It contains 60,000 training and 10,000 testing samples, each a $28 \times 28$ pixel image. The dataset offers a diverse range of clothing categories, making it suitable for evaluating algorithms and models for tasks such as image recognition, object detection, and fashion-related applications. FashionMNIST provides a challenging yet realistic dataset for researchers and practitioners to explore and develop innovative solutions in computer vision.

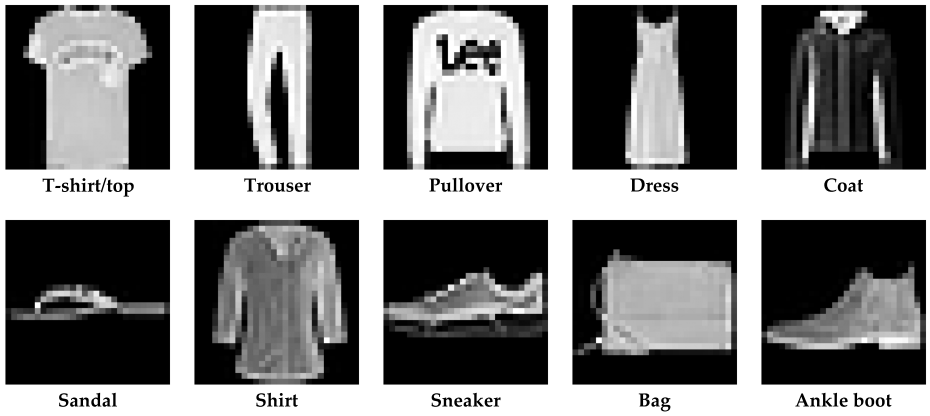


**Figure 5.** The FashionMNIST dataset consists of 10 classes of monochrome clothing items and is divided into 60,000 images for the training set and 10,000 images for the test set.

### *3.2. LeNet-5 and AlexNet*

LeNet-5 is a classic CNN architecture developed by Yann LeCun [11]. It was explicitly designed for handwritten digit recognition and played a crucial role in the early advancements of deep learning. LeNet-5 is composed of convolutional, pooling, and fully connected layers (see Figure 6). The convolutional layers extract features from the input images using convolutional filters. The pooling layers reduce the dimensionality of the extracted features while preserving their essential information. Finally, the fully connected layers classify the features and produce the output predictions. LeNet-5 revolutionized the field of computer vision by demonstrating the effectiveness of CNNs for image classification tasks. Since then, it has served as a foundational model for developing more advanced CNN architectures and has found applications in various domains, including character recognition, object detection, and facial recognition.

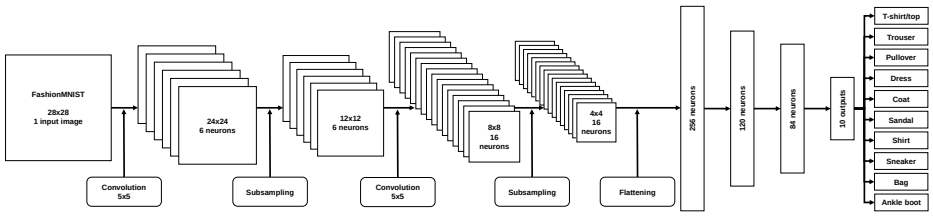


**Figure 6.** The LeNet-5 architecture, applied to the FashionMNIST dataset, used for the research.

AlexNet [6] is a nine-layer DNN composed of six convolutional layers and three fully-connected layers. It represents the reference model for deep CNNs, where stacking several layers resulted in significant performance improvements compared to shallower CNNs. A sequence of several convolutional layers can learn high-level features from the inputs that are used by fully connected layers to generate the output predictions.

### *3.3. Spiking-LeNet-5, Spiking-AlexNet and Norse*

Spiking-LeNet-5 [46–49] is an extension of the LeNet-5 CNN architecture that incorporates the principles of SNNs [50]. It is specifically designed to process temporal data encoded as spike trains, mimicking the behavior of biological neurons. Unlike the traditional LeNet-5, which operates on static input values, Spiking-LeNet-5 receives input spikes as a sequence of binary tensors. It utilizes specialized spiking neuron models, such as the leaky integrate-and-fire (LIF) neuron, to simulate the firing behavior of biological neurons [51]. The temporal dimension introduced by spike encoding allows Spiking-LeNet-5 to capture the dynamics and temporal dependencies present in the data. This enables the network to learn and recognize patterns over time, making it suitable for tasks involving temporal data, such as event-based vision, audio processing, and other time-dependent applications. Spiking-LeNet-5 combines the power of traditional CNNs with the temporal processing capabilities of SNNs, opening up new possibilities for advanced SNN architectures. Similarly, Spiking-AlexNet [52] extends AlexNet by incorporating the principles of SNNs, such as spike trains and LIF neurons.

The LIF parameters [53] in Norse are specific settings that define the behavior of LIF neurons in SNNs. These parameters include:

- $\tau_{syn}^{-1}$—represents the inverse of the synaptic time constant. It determines the rate at which the synaptic input decays over time;
- $\tau_{mem}^{-1}$—represents the inverse of the membrane time constant. This parameter influences the rate at which the neuron's membrane potential decays without input;
- $v_{leak}$—specifies the leak potential of the neuron. It is the resting potential of the neuron's membrane when there is no synaptic input or other stimuli;
- $v_{th}$—defines the threshold potential of the neuron. The neuron generates an action potential when the membrane potential reaches or exceeds this threshold;
- $v_{reset}$—represents the reset potential of the neuron. After firing an action potential, the membrane potential is reset to this value.

These parameters play a crucial role in shaping the dynamics of the LIF neuron in the SNN. They determine how the neuron integrates and responds to incoming synaptic input and when it generates an action potential. The specific values of these parameters can be adjusted to achieve desired behavior and control the firing rate and responsiveness of the neuron within the network.

SNNs also require an encoder because they operate on temporal data represented as spikes. Because most ML datasets do not include any temporal encoding, it is necessary to add an encoding step to provide the required temporal dimension. The encoder transforms the input data into sequences of spikes, which are then processed by the SNN as tensors containing binary values. The constant-current LIF encoder is an encoding method used in the Norse library to transform input data into sparse spikes. This encoding technique

converts the constant input current into constant voltage spikes. During a specified time interval, known as $seq_{length}$, spikes are simulated based on the input current. This encoding allows Norse to operate on sparse input data in a sequence of binary tensors, which the SNN can efficiently process.

### *3.4. HE Parameters and Pyfhel*

The HE process, implemented in the Pyfhel library, allows computations on encrypted data without decryption, ensuring data privacy and security [54,55]. Pyfhel is built on the BFV scheme, a fully HE scheme.

The encryption process in the BFV scheme involves transforming plaintext data into ciphertext using a public key [56]. The computations can be directly conducted on the ciphertext, preserving the confidentiality of the underlying plaintext [57]. The BFV scheme supports various mathematical operations on encrypted data, such as addition and multiplication. These operations can be performed on ciphertexts without decryption, enabling computations on sensitive data while maintaining its privacy [58].

The BFV scheme relies on three key parameters:

- $m$—represents the polynomial modulus degree, influencing the encryption scheme's computational capabilities and security level;
- $t$—denotes the plaintext modulus and determines the size and precision of the encrypted plaintext values;
- $q$—represents the ciphertext modulus, determining the size of the encrypted ciphertext values and affecting the security and computational performance of the encryption scheme.

A balance between security and computational efficiency in HE computations can be achieved by selecting appropriate values for these parameters. Pyfhel provides a convenient interface to work with the BFV scheme, allowing for data encryption, computation, and decryption while maintaining privacy and confidentiality.

Another critical parameter is the noise budget (NB), which refers to the maximum amount of noise or error that can be introduced during the encryption and computation process without affecting the correctness of the results. When performing computations on encrypted data, operations such as additions and multiplications can accumulate noise, deleting the decrypted results' accuracy. The NB represents a limit on how much noise can be tolerated before the decrypted results become unreliable. The NB needs to be carefully managed and monitored throughout the computation process to ensure the security and correctness of the encrypted computations.

## 4. Results and Discussion

The experiments are divided into several parts to obtain accurate results:

- Training of the LeNet-5, AlexNet, Spiking-LeNet-5, and Spiking-AlexNet models on the training set of the FashionMNIST dataset;
- Validating the models on the test set of the same dataset;
- Creating encrypted models based on the previously trained models [59];
- Encrypting the test set;
- Evaluating the encrypted images on the encrypted LeNet-5, AlexNet, Spiking-LeNet-5, and Spiking-AlexNet models.

### *4.1. Training Phase*

For the training phase, optimal parameters were set to increase accuracy. The best learning rate was found using the learning rate finder technique [60], whereas the number of epochs was chosen based on early stopping to prevent overfitting [61]. Table 1 reports all the parameters chosen for the training phase.

**Table 1.** Training phase parameters.

| Parameters | LeNet-5 | Spiking-LeNet-5 | AlexNet | Spiking-AlexNet |
|---|---|---|---|---|
| Learning Rate | 0.001 | 0.001 | 0.0001 | 0.0001 |
| Epochs | 20 | 20 | 20 | 20 |
| Optimizer [62] | Adam | Adam | Adam | Adam |
| Loss [63] | Cross Entropy | Negative Log-Likelihood | Cross Entropy | Negative Log-Likelihood |
| $seq_{length}$ | - | 30 | - | 30 |
| $\tau_{syn}^{-1}$ | - | 200 | - | 200 |
| $\tau_{mem}^{-1}$ | - | 100 | - | 100 |
| $v_{leak}$ | - | 0 | - | 0 |
| $v_{th}$ | - | 0.5 | - | 0.5 |
| $v_{reset}$ | - | 0 | - | 0 |
| Encoder | - | Constant Current LIF | - | Constant Current LIF |

Figure 7 shows the accuracy and loss during training, comparing the LeNet-5 CNN with Spiking-LeNet-5 and their respective validation values at each epoch.

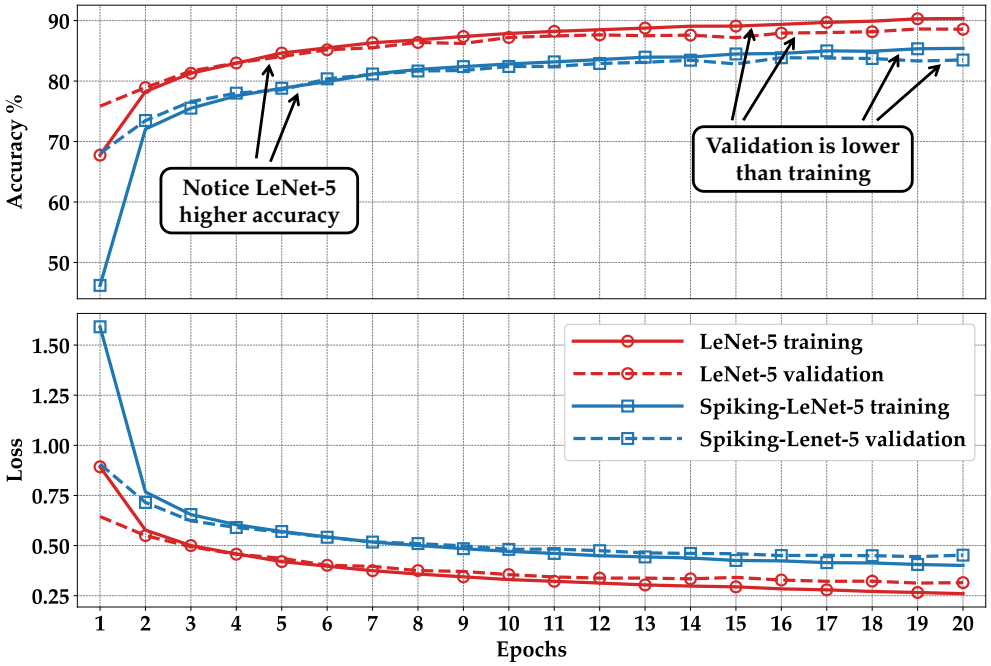


**Figure 7.** Accuracy and loss during training and validation of LeNet-5 and Spiking-LeNet-5 for the FashionMNIST dataset. The figure shows accuracy and loss values across different training epochs.

Note that Spiking-LeNet-5 has slightly lower accuracy than (non-spiking) LeNet-5 due to the intrinsic complexity of the model itself, and its computational time is, on average, equal to that of LeNet-5 multiplied by the value of the $seq_{length}$.

### *4.2. Encryption*

It is necessary to determine the three fundamental parameters that define a BFV HE scheme to proceed with image encryption: $m$, $t$, and $q$.

The $m$ parameter is chosen as a power of two and is directly proportional to the amount of NB. Values that are too small would be insecure, whereas values that are too large would make the computation too complex. Generally, $m$ is never less than 1024, and in our specific case, we observe that values of 2048 or higher do not influence the results but incur in exponentially longer computation time. For these reasons, we chose to keep the parameter $m$ fixed at 1024.

The $t$ parameter can also vary, and low values do not allow for proper encryption, whereas excessively high values degrade the result due to computational complexity. In our case, we evaluated the results over values ranging from 10 to 5000.

The $q$ parameter is closely related to the $m$ parameter in determining the NB. Hence, it is automatically calculated by the Pyfhel library to achieve proper encryption.

With the hardware at our disposal (Tesla P100-PCIE GPU, Intel(R) Xeon(R) Gold 6134 @ 3.20 GHz CPU, and 100 GB of RAM), it took approximately 30 s to encrypt each image and an additional 30 s to evaluate encrypted LeNet-5. However, for evaluation on encrypted Spiking-LeNet-5, it took around 15 min due to the $seq_{length}$ parameter equal to 30. For a clearer visualization, Table 2 shows a comparison of the computation times for each image along with estimates for other models: AlexNet [6], VGG-16 [64], and ResNet-50 [65]. These long execution times are aligned with the recent trend in the community that demands to build specialized accelerators for HE. A popular example is represented by the data protection in virtual environments (DPRIVE) challenge, used by DARPA to sponsor organizations that pursue R&D of HE hardware [66–68].

**Table 2.** Execution time for each image reported in seconds for each model. The total encrypted execution is broken down into encryption and processing time of encrypted data. The long processing times of encrypted data are due to the complexity of the encrypted computations.

| Time (seconds) | LeNet-5 | Spiking-LeNet-5 | AlexNet | Spiking-AlexNet | VGG-16 | Spiking-VGG-16 | ResNet-50 | Spiking-ResNet-50 |
|---|---|---|---|---|---|---|---|---|
| Normal execution (unencrypted) | 0.03 | 1 | 30 | 1000 | 70 | 2300 | 10 | 300 |
| Encrypted execution | 31 | 930 | 30,060 | 901,800 | 70,140 | 2,104,200 | 10,020 | 300,600 |
| Encryption | 1 | 30 | 60 | 1800 | 140 | 4200 | 20 | 600 |
| Processing time of encrypted data | 30 | 900 | 30,000 | 900,000 | 70,000 | 2,100,000 | 10,000 | 300,000 |

### 4.3. Evaluation

In Figures 8–11, we can observe the results of encryption compared to the standard ones, along with the correct labels as the parameter $t$ varies.

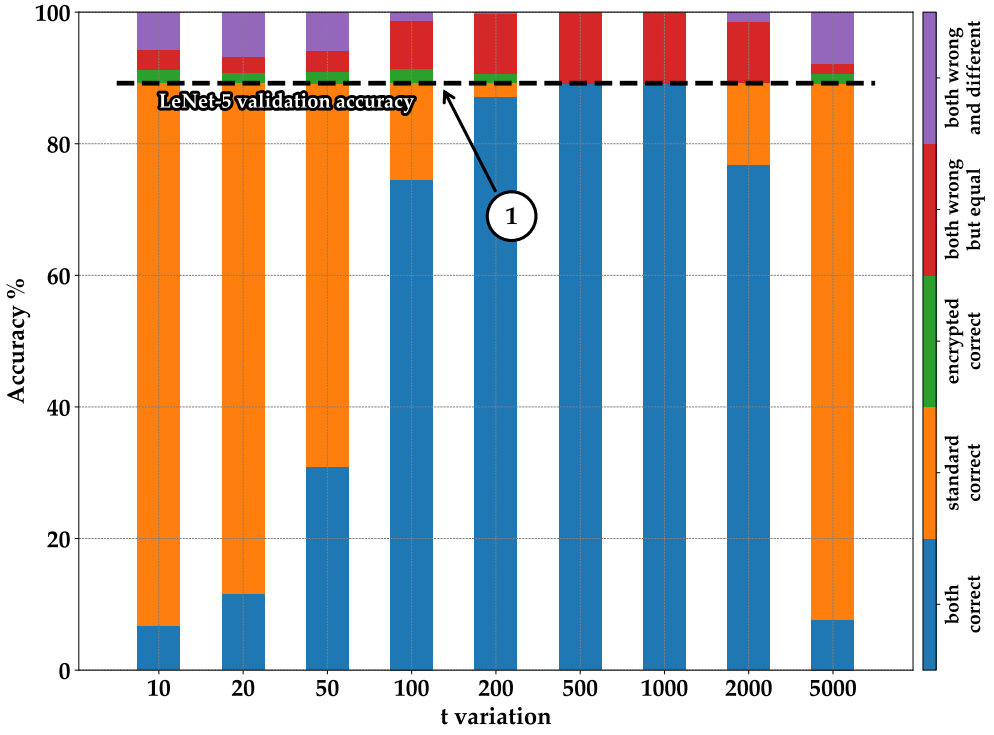


**Figure 8.** FashionMNIST accuracy on LeNet-5 for $t$ variation.

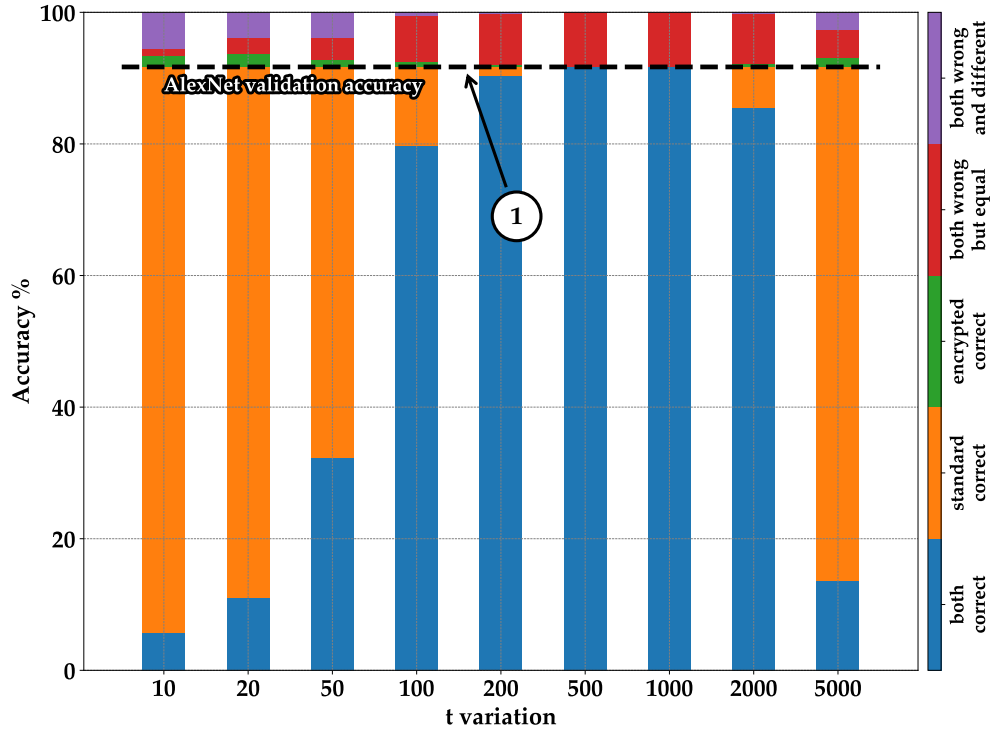


**Figure 9.** FashionMNIST accuracy on AlexNet for *t* variation.

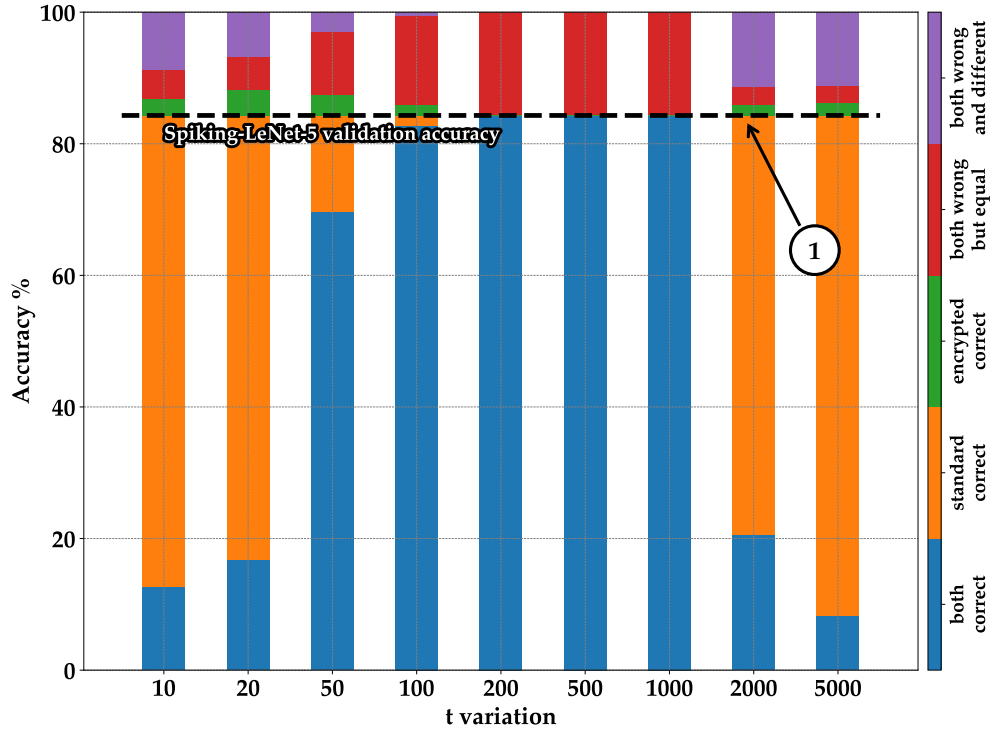


**Figure 10.** FashionMNIST accuracy on Spiking-LeNet-5 for *t* variation.

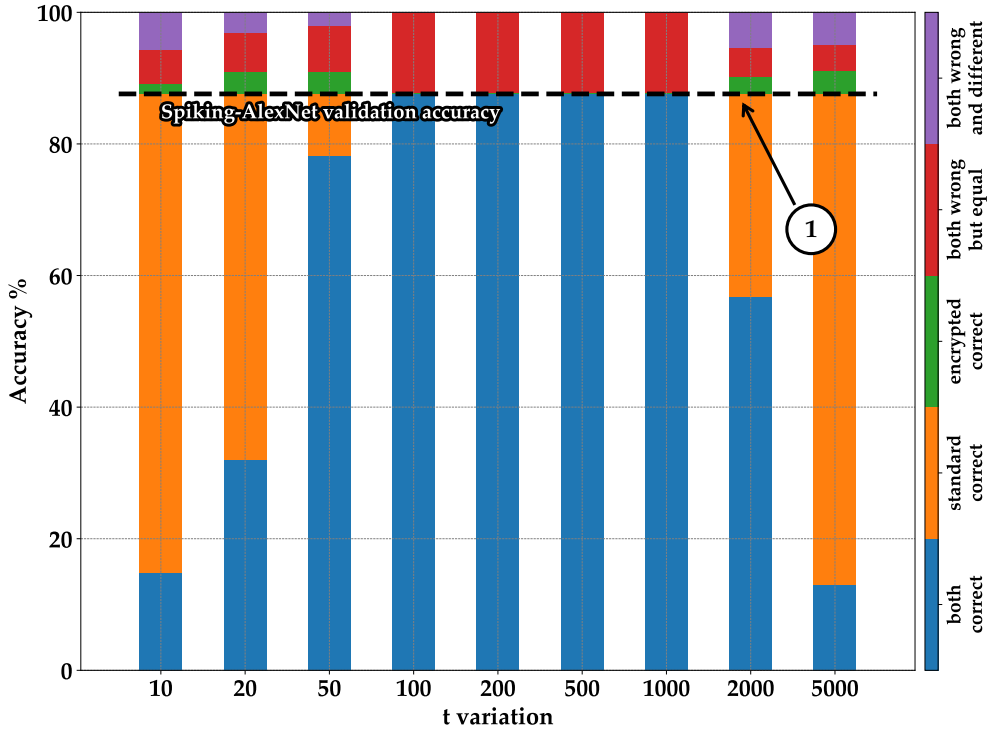


**Figure 11.** FashionMNIST accuracy on Spiking-AlexNet for $t$ variation.

The various parts of the bars in the figures are divided as follows:

- **Blue—both correct:** indicates the number of images classified correctly in both the standard and encrypted executions;
- **Orange—standard correct:** represents the case where images are classified correctly in the standard execution but not in the encrypted one. It can be observed that by summing the blue and orange columns, we always obtain the same result: the accuracy of validation during training (see pointer ①—Figures 8–11);
- **Green—encrypted correct:** represents images classified correctly in the encrypted case but not in the standard one. It can be noticed that the percentages are generally low; this is because the encrypted model mistakenly classified the images differently from the standard model, but by chance, it happened to choose the correct label. Therefore, this column part does not represent a valid statistical case but rather randomness;
- **Red—both wrong but equal:** indicates cases where the encrypted model was classified identically to the standard one but did not classify the correct label. This part is essential, as it shows the encrypted model working correctly by emulating the standard model, even though the classification is incorrect overall;
- **Purple—both wrong and different:** shows cases where the encrypted model made mistakes by not producing the same result as the standard model, and the standard model also made mistakes by not classifying correctly.

It can be noticed that for both low and high values of $t$, the results degrade rapidly. For a better understanding, let us compare LeNet-5 with Spiking-LeNet-5 by looking at Figures 12 and 13, and AlexNet with Spiking-AlexNet in Figures 14 and 15, where the accuracies are graphically displayed as $t$ varies.

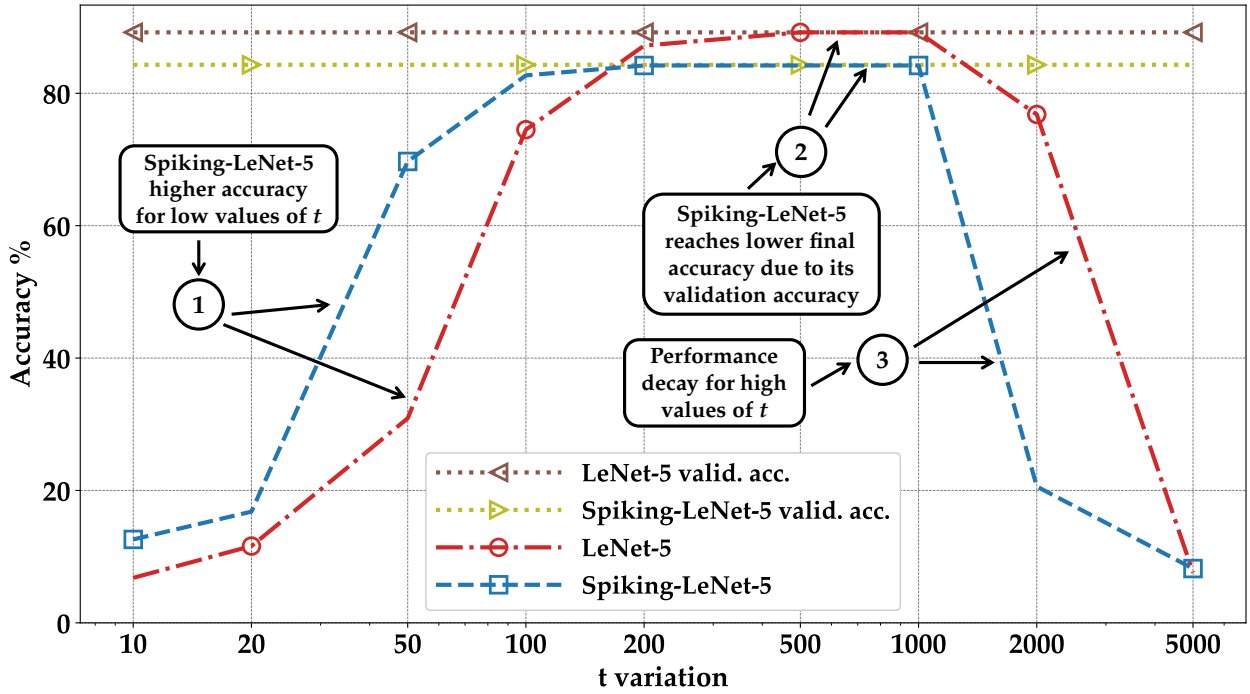


**Figure 12.** Comparison of FashionMNIST accuracy between LeNet-5 and Spiking-LeNet-5 for $t$ variations when both standard and encrypted versions classified correctly.

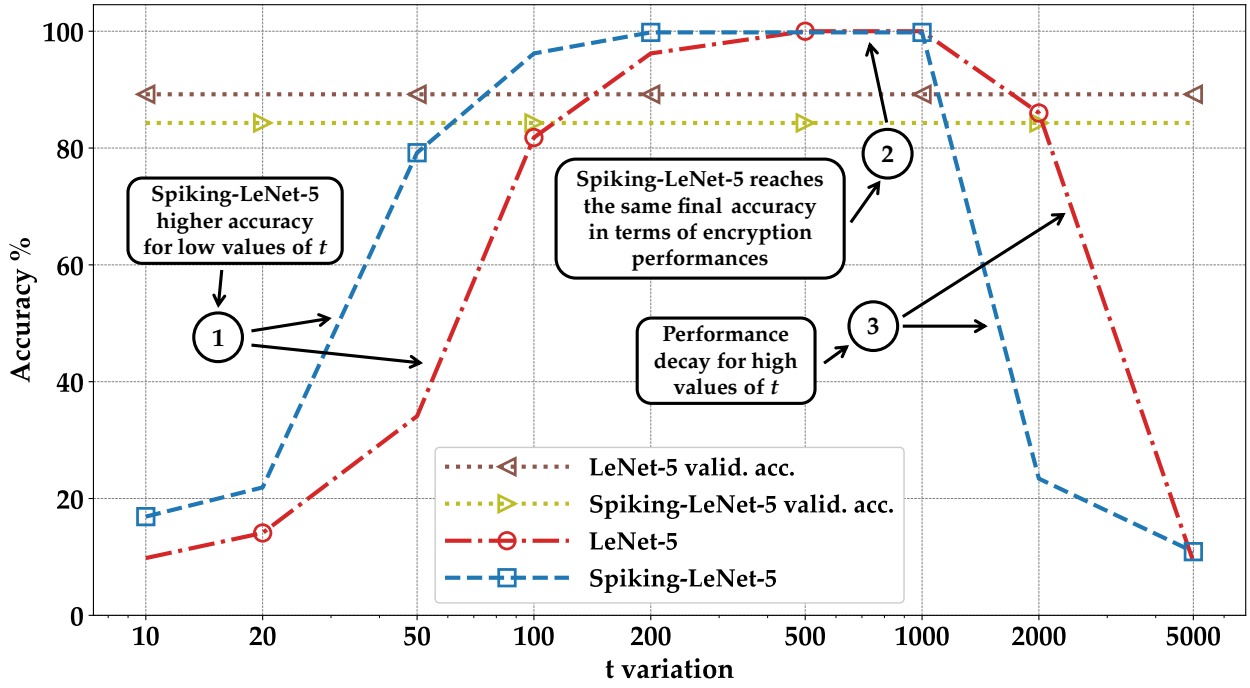


**Figure 13.** Comparison of FashionMNIST accuracy between LeNet-5 and Spiking-LeNet-5 for $t$ when the standard and encrypted versions coincide in both correct and incorrect classification.

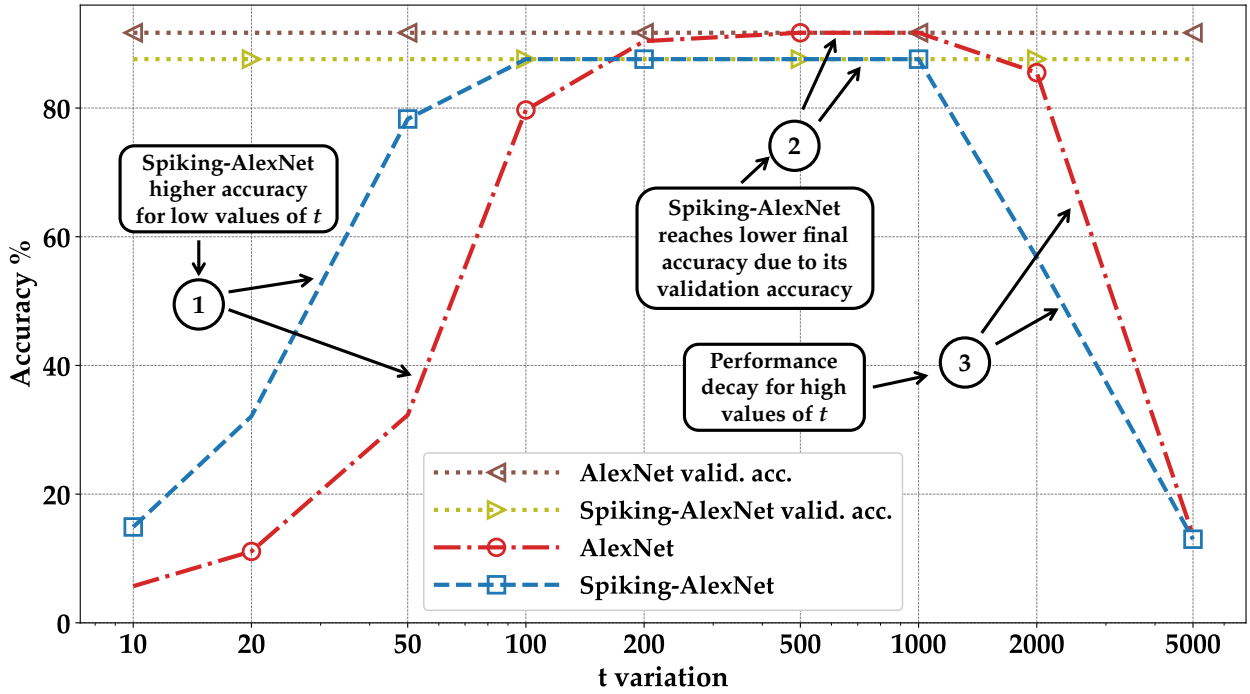


**Figure 14.** Comparison of FashionMNIST accuracy between AlexNet and Spiking-AlexNet for $t$ variations when both standard and encrypted versions classified correctly.

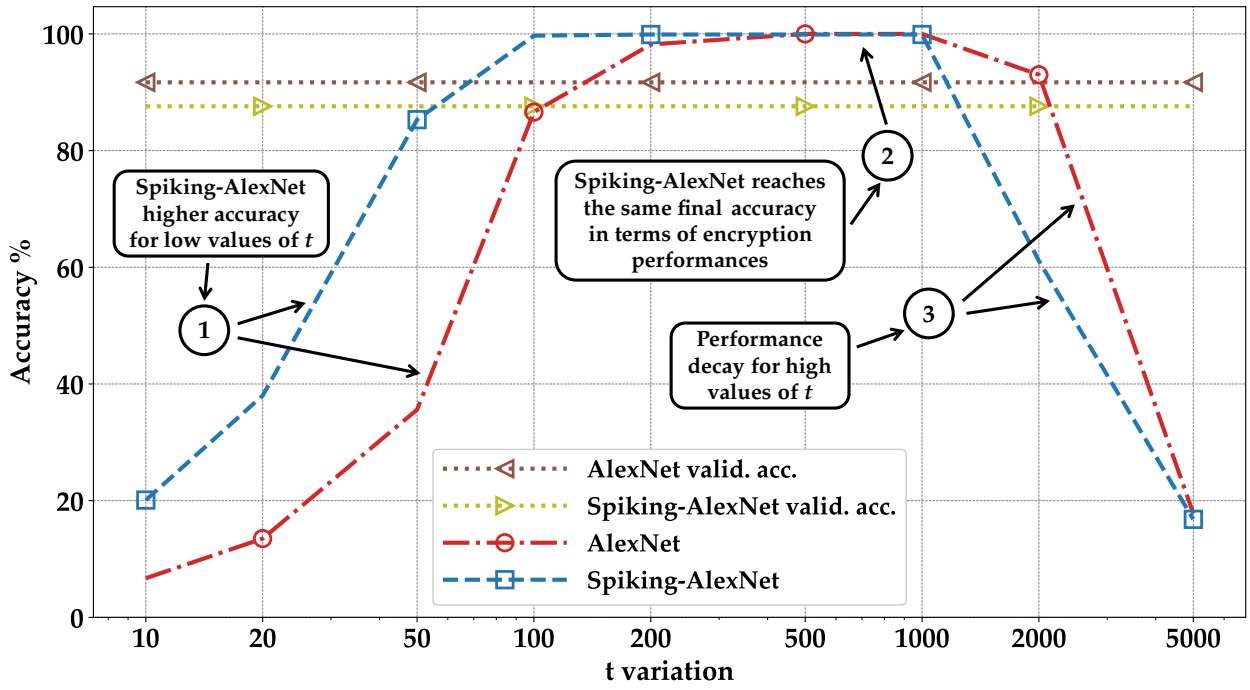


**Figure 15.** Comparison of FashionMNIST accuracy between AlexNet and Spiking-AlexNet for $t$ when the standard and encrypted versions coincide in both correct and incorrect classification.

In Figures 12 and 14, we compared the LeNet-5, Spiking-LeNet-5, AlexNet, and Spiking-AlexNet models in the case where both the standard and encrypted models were correct, representing the graphical representation of the blue parts of Figures 8–11. As can be seen, the Spiking-LeNet-5 version achieves acceptable levels of accuracy much earlier than LeNet-5, even with low values of $t$ (see pointer ①—Figure 12). For instance, when $t = 50$, Spiking-LeNet-5 achieves about 40% higher accuracy than LeNet-5. However, the

final accuracy of the Spiking-LeNet-5 model is slightly lower than that of LeNet-5 (see pointer ②—Figure 12); this can be attributed to the fact that the Spiking-LeNet-5 model itself had lower validation accuracy compared to LeNet-5, as shown in Figure 7. Similar observations can be derived by comparing AlexNet with Spiking-AlexNet. Spiking-AlexNet reaches higher accuracy than the AlexNet for low values of $t$ (see pointer ①—Figure 14), but for larger $t$, the accuracy of AlexNet is slightly higher than that of Spiking-AlexNet (see pointer ②—Figure 14).

On the contrary, in Figures 13 and 15, we compared the sums of the blue and red parts from Figures 8–11. In this manner, we can observe all the cases where the encrypted version produced the same result as the standard one, even if it was incorrect (see pointer ②—Figures 13 and 15). From this graph, we can notice that the encrypted version of the Spiking-LeNet-5 model performs better than the encrypted LeNet-5, and the encrypted Spiking-AlexNet performs better than the encrypted AlexNet. The SNNs achieve valid results with lower values of $t$ (see pointer ①—Figures 13 and 15) and higher overall accuracy. For excessively high values of $t$, the results degrade for both the DNN and SNN models due to the increased computational complexity, which hinders the attainment of acceptable outputs (see pointer ③—Figures 12–15).

## 5. Conclusions

In this work, we have demonstrated how SNNs can be a crucial factor in the development of future private and secure networks. Despite the increased time requirement, SNNs offer higher reliability, and further research can potentially reduce the time differences between DNNs and SNNs. The use of encryption systems such as HE is now more important than ever, considering the vast amount of data being exchanged worldwide. In this research, we have successfully shown how complete encryption systems can be applied to complex models, both CNNs and SNNs, ensuring correct final results without the possibility of decoding during the intermediate process, and how these results differ for CNNs and SNNs. Our work represents the first proof-of-concept that demonstrates the applicability of HE schemes to SNNs. In future works, we plan to design acceleration techniques for encrypted SNNs and extend the experiment set with deeper networks.

**Author Contributions:** Conceptualization, F.N., R.C., A.M., M.M. and M.S.; methodology, F.N., R.C., A.M., M.M. and M.S.; software, F.N., R.C. and A.M.; validation, F.N., R.C. and A.M.; formal analysis, F.N., R.C. and A.M.; investigation, F.N., R.C. and A.M.; resources, F.N., R.C. and A.M.; data curation, F.N., R.C. and A.M.; writing—original draft preparation, F.N., R.C. and A.M.; writing—review and editing, F.N., R.C., A.M., M.M. and M.S.; visualization, F.N., R.C. and A.M.; supervision, F.N., A.M., M.M. and M.S.; project administration, M.M. and M.S.; funding acquisition, M.M. and M.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work has been supported in part by the Doctoral College Resilient Embedded Systems, which is run jointly by the TU Wien's Faculty of Informatics and the UAS Technikum Wien. This work was also supported in parts by the NYUAD Center for Cyber Security (CCS), funded by Tamkeen under the NYUAD Research Institute Award G1104, and the Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010.

**Data Availability Statement:** Open-source framework: https://github.com/Nikfam/SpyKing (accessed on 1 August 2023).

**Conflicts of Interest:** The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| ML | Machine Learning |
| DNNs | Deep Neural Networks |
| SNNs | Spiking Neural Networks |
| HE | Homomorphic Encryption |
| PHE | Partially Homomorphic Encryption |
| SHE | Somewhat Homomorphic Encryption |
| FHE | Fully Homomorphic Encryption |
| BFV | Brakerski/Fan-Vercauteren |
| CNNs | Convolutional Neural Networks |
| LIF | Leaky Integrate-and-Fire |
| NB | Noise Budget |

## References


1. Capra, M.; Bussolino, B.; Marchisio, A.; Masera, G.; Martina, M.; Shafique, M. Hardware and Software Optimizations for Accelerating Deep Neural Networks: Survey of Current Trends, Challenges, and the Road Ahead. *IEEE Access* **2020**, *8*, 225134–225180. [CrossRef]
2. Dave, S.; Marchisio, A.; Hanif, M.A.; Guesmi, A.; Shrivastava, A.; Alouani, I.; Shafique, M. Special Session: Towards an Agile Design Methodology for Efficient, Reliable, and Secure ML Systems. In Proceedings of the 40th IEEE VLSI Test Symposium, VTS 2022, San Diego, CA, USA, 25–27 April 2022; pp. 1–14. [CrossRef]
3. Shafique, M.; Marchisio, A.; Putra, R.V.W.; Hanif, M.A. Towards Energy-Efficient and Secure Edge AI: A Cross-Layer Framework ICCAD Special Session Paper. In Proceedings of the IEEE/ACM International Conference on Computer Aided Design, ICCAD 2021, Munich, Germany, 1–4 November 2021; pp. 1–9. [CrossRef]
4. Simeone, O.; Rajendran, B.; Grüning, A.; Eleftheriou, E.; Davies, M.; Denève, S.; Huang, G. Learning Algorithms and Signal Processing for Brain-Inspired Computing. *IEEE Signal Process. Mag.* **2019**, *36*, 12–15. [CrossRef]
5. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
6. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
7. von Kügelgen, J. On Artificial Spiking Neural Networks: Principles, Limitations and Potential. Master's Thesis, University of Barcelona, Barcelona, Spain, 2017.
8. Diamond, A.; Nowotny, T.; Schmuker, M. Comparing Neuromorphic Solutions in Action: Implementing a Bio-Inspired Solution to a Benchmark Classification Task on Three Parallel-Computing Platforms. *Front. Neurosci.* **2016**, *9*, 491. [CrossRef]
9. Barni, M.; Orlandi, C.; Piva, A. A privacy-preserving protocol for neural-network-based computation. In Proceedings of the 8th workshop on Multimedia & Security, MM&Sec 2006, Geneva, Switzerland, 26–27 September 2006; pp. 146–151. [CrossRef]
10. Fan, J.; Vercauteren, F. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch.* **2012**, *2012*, 144.
11. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
12. Johnson, J.D.; Li, J.; Chen, Z. Reinforcement Learning: An Introduction: R.S. Sutton, A.G. Barto, MIT Press: Cambridge, MA, USA, 1998; p. 322, ISBN 0-262-19398-1. *Neurocomputing* **2000**, *35*, 205–206. [CrossRef]
13. Ponulak, F.; Kasiński, A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiol. Exp.* **2011**, *71*, 409–433.
14. Paugam-Moisy, H.; Bohté, S.M. Computing with Spiking Neuron Networks. In *Handbook of Natural Computing*; Rozenberg, G., Bäck, T., Kok, J.N., Eds.; Springer: Berlin, Germany, 2012; pp. 335–376. . [CrossRef]
15. Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S.R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Netw.* **2019**, *111*, 47–63. [CrossRef]
16. Marchisio, A.; Nanfa, G.; Khalid, F.; Hanif, M.A.; Martina, M.; Shafique, M. Is Spiking Secure? A Comparative Study on the Security Vulnerabilities of Spiking and Deep Neural Networks. In Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
17. Marchisio, A.; Pira, G.; Martina, M.; Masera, G.; Shafique, M. R-SNN: An Analysis and Design Methodology for Robustifying Spiking Neural Networks against Adversarial Attacks through Noise Filters for Dynamic Vision Sensors. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, 27 September–1 October 2021; pp. 6315–6321. [CrossRef]
18. El-Allami, R.; Marchisio, A.; Shafique, M.; Alouani, I. Securing Deep Spiking Neural Networks against Adversarial Attacks through Inherent Structural Parameters. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, 1–5 February 2021; pp. 774–779. [CrossRef]
19. Kim, Y.; Chough, J.; Panda, P. Beyond classification: Directly training spiking neural networks for semantic segmentation. *Neuromorph. Comput. Eng.* **2022**, *2*, 44015. [CrossRef]

20. Meftah, B.; Lézoray, O.; Chaturvedi, S.; Khurshid, A.A.; Benyettou, A. Image Processing with Spiking Neuron Networks. In *Artificial Intelligence, Evolutionary Computing and Metaheuristics—In the Footsteps of Alan Turing*; Yang, X., Ed.; Springer: Berlin, Germany, 2013; Volume 427, pp. 525–544. [CrossRef]
21. Viale, A.; Marchisio, A.; Martina, M.; Masera, G.; Shafique, M. CarSNN: An Efficient Spiking Neural Network for Event-Based Autonomous Cars on the Loihi Neuromorphic Research Processor. In Proceedings of the International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, 18–22 July 2021; pp. 1–10. [CrossRef]
22. Cordone, L.; Miramond, B.; Thiérion, P. Object Detection with Spiking Neural Networks on Automotive Event Data. In Proceedings of the International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, 18—23 July 2022; pp. 1–8. [CrossRef]
23. Viale, A.; Marchisio, A.; Martina, M.; Masera, G.; Shafique, M. LaneSNNs: Spiking Neural Networks for Lane Detection on the Loihi Neuromorphic Processor. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, 23–27 October 2022; pp. 79–86. [CrossRef]
24. Massa, R.; Marchisio, A.; Martina, M.; Shafique, M. An Efficient Spiking Neural Network for Recognizing Gestures with a DVS Camera on the Loihi Neuromorphic Processor. In Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, UK, 19–24 July 2020; pp. 1–9. [CrossRef]
25. Indiveri, G.; Sandamirskaya, Y. The Importance of Space and Time for Signal Processing in Neuromorphic Agents: The Challenge of Developing Low-Power, Autonomous Agents That Interact With the Environment. *IEEE Signal Process. Mag.* **2019**, *36*, 16–28. [CrossRef]
26. Lee, J.; Delbrück, T.; Pfeiffer, M. Training Deep Spiking Neural Networks using Backpropagation. *arXiv* **2016**, arXiv:1608.08782.
27. Lee, C.; Sarwar, S.S.; Roy, K. Enabling Spike-based Backpropagation in State-of-the-art Deep Neural Network Architectures. *arXiv* **2019**, arXiv:1903.06379.
28. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178. [CrossRef]
29. Orlandi, C.; Piva, A.; Barni, M. Oblivious Neural Network Computing via Homomorphic Encryption. *EURASIP J. Inf. Secur.* **2007**, *2007*, 037343. [CrossRef]
30. Stehlé, D.; Steinfeld, R.; Tanaka, K.; Xagawa, K. Efficient Public Key Encryption Based on Ideal Lattices. In Proceedings of the Advances in Cryptology-ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, 6–10 December 2009; Volume 5912, pp. 617–635. [CrossRef]
31. Damgård, I.; Jurik, M. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In Proceedings of the Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Republic of Korea, 13–15 February 2001; Volume 1992, pp. 119–136. [CrossRef]
32. Rivest, R.L.; Dertouzos, M.L. *On Data Banks and Privacy Homomorphisms*; Academic Press, Inc.: Cambridge, MA, USA, 1978.
33. Bos, J.W.; Lauter, K.E.; Loftus, J.; Naehrig, M. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. *IACR Cryptol. ePrint Archive* **2013**, *2013*, 75.
34. Chabanne, H.; de Wargny, A.; Milgram, J.; Morel, C.; Prouff, E. Privacy-Preserving Classification on Deep Neural Network. *IACR Cryptol. ePrint Arch.* **2017**, *2017*, 35.
35. Falcetta, A.; Roveri, M. Privacy-Preserving Deep Learning With Homomorphic Encryption: An Introduction. *IEEE Comput. Intell. Mag.* **2022**, *17*, 14–25. [CrossRef]
36. Brakerski, Z.; Vaikuntanathan, V. Efficient Fully Homomorphic Encryption from (Standard) $\mathsf{LWE}$. *SIAM J. Comput.* **2014**, *43*, 831–871. [CrossRef]
37. Gentry, C. A Fully Homomorphic Encryption Scheme. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2009.
38. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. Fully Homomorphic Encryption without Bootstrapping. *IACR Cryptol. ePrint Arch.* **2011**, *TR11*, 277.
39. Boneh, D.; Goh, E.; Nissim, K. Evaluating 2-DNF Formulas on Ciphertexts. In Proceedings of the Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, 10–12 February 2005; Volume 3378, pp. 325–341. [CrossRef]
40. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
41. Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
42. Patiño-Saucedo, A.; Rostro-González, H.; Serrano-Gotarredona, T.; Linares-Barranco, B. Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the SpiNNaker neuromorphic platform. *Neural Netw.* **2020**, *121*, 319–328. [CrossRef] [PubMed]
43. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; pp. 8024–8035.

44. Ibarrondo, A.; Viand, A. Pyfhel: PYthon For Homomorphic Encryption Libraries. In Proceedings of the WAHC '21: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Virtual Event, Republic of Korea, 15 November 2021; pp. 11–16. [CrossRef]
45. Pehle, C.G.; Pedersen, J.E. Norse—A Deep Learning Library for Spiking Neural Networks. 2021. Available online: https://norse.ai/docs/ (accessed on 1 August 2023).
46. Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; Roy, K. Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *arXiv* **2018**, arXiv:1802.02627.
47. Izhikevich, E.M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* **2003**, *14*, 1569–1572. [CrossRef]
48. Roy, K.; Jaiswal, A.R.; Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature* **2019**, *575*, 607–617. [CrossRef]
49. Han, B.; Roy, K. Deep Spiking Neural Network: Energy Efficiency Through Time Based Coding. In Proceedings of the Computer Vision—ECCV 2020—16th European Conference, Glasgow, UK, 23–28 August 2020; Volume 12355, pp. 388–404. [CrossRef]
50. Zenke, F.; Ganguli, S. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Comput.* **2018**, *30*, 1514. [CrossRef] [PubMed]
51. Ponulak, F.; Kasinski, A.J. Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting. *Neural Comput.* **2010**, *22*, 467–510. [CrossRef]
52. Guo, W.; Fouda, M.E.; Eltawil, A.; Salama, K. Efficient training of spiking neural networks with temporally-truncated local backpropagation through time. *Front. Neurosci.* **2023**, *17*, 1047008. [CrossRef] [PubMed]
53. Chen, H.; Laine, K.; Player, R. Simple Encrypted Arithmetic Library—SEAL v2.1. *IACR Cryptol. ePrint Arch.* **2017**, 224. Available online: https://eprint.iacr.org/2017/224 (accessed on 1 August 2023).
54. Papernot, N.; McDaniel, P.D.; Sinha, A.; Wellman, M.P. Towards the Science of Security and Privacy in Machine Learning. *arXiv* **2016**, arXiv:1611.03814.
55. Yao, A.C. Protocols for Secure Computations (Extended Abstract). In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, Chicago, IL, USA, 3–5 November 1982; pp. 160–164. [CrossRef]
56. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Proceedings of the Advances in Cryptology—EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; pp. 223–238. [CrossRef]
57. Disabato, S.; Falcetta, A.; Mongelluzzo, A.; Roveri, M. A Privacy-Preserving Distributed Architecture for Deep-Learning-as-a-Service. In Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
58. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.E.; Naehrig, M.; Wernsing, J. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, 19–24 June 2016; Volume 48, pp. 201–210.
59. Kim, Y.; Venkatesha, Y.; Panda, P. PrivateSNN: Privacy-Preserving Spiking Neural Networks. In Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022, Virtual Event, 22 February–1 March 2022; pp. 1192–1200.
60. Smith, L.N. Cyclical Learning Rates for Training Neural Networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472. [CrossRef]
61. Rice, L.; Wong, E.; Kolter, J.Z. Overfitting in adversarially robust deep learning. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Virtual Event, 13–18 July 2020; Volume 119, pp. 8093–8104.
62. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
63. Janocha, K.; Czarnecki, W.M. On Loss Functions for Deep Neural Networks in Classification. *arXiv* **2017**, arXiv:1702.05659.
64. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
65. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
66. Cammarota, R. Intel HERACLES: Homomorphic Encryption Revolutionary Accelerator with Correctness for Learning-oriented End-to-End Solutions. In Proceedings of the 2022 on Cloud Computing Security Workshop, CCSW 2022, Los Angeles, CA, USA, 7 November 2022; p. 3. [CrossRef]

67. Badawi, A.A.; Bates, J.; Bergamaschi, F.; Cousins, D.B.; Erabelli, S.; Genise, N.; Halevi, S.; Hunt, H.; Kim, A.; Lee, Y.; et al. OpenFHE: Open-Source Fully Homomorphic Encryption Library. In Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Los Angeles, CA, USA, 7 November 2022; pp. 53–63. [CrossRef]
68. Cousins, D.B.; Polyakov, Y.; Badawi, A.A.; French, M.; Schmidt, A.G.; Jacob, A.P.; Reynwar, B.; Canida, K.; Jaiswal, A.R.; Mathew, C.; et al. TREBUCHET: Fully Homomorphic Encryption Accelerator for Deep Computation. *arXiv* **2023**, arXiv:2304.05237.