Privacy-Preserving Machine Learning in IoT: A Study of Data Obfuscation Methods

Yonan Yonan[1], Mohammad O. Abdullah[1], Felix Nilsson[2], Mahdi Fazeli[1], Ahmad Patooghy[3] and Slawomir Nowaczyk[1]

[1]*School of Information Technology, Halmstad University, Sweden*

[2]*HMS Industrial Networks, Halmstad, Sweden*

[3]*Department of Computer Systems Technology, North Carolina A&T State University, NC, U.S.A.*

Keywords: Machine Learning, Neural Networks, Data Obfuscation, Image Classification, IoT, Cloud Computing, Resource Constrained Devices.

Abstract: In today's interconnected digital world, ensuring data privacy is critical, particularly for neural networks operating remotely in the age of the Internet of Things (IoT). This paper tackles the challenge of data privacy preservation in IoT environments by investigating Utility-Preserving Data Transformation (UPDT) methods, which aim to transform data in ways that reduce or eliminate sensitive information while retaining its utility for analytical tasks. UPDT methods aim to balance privacy preservation and utility in data analytics. This study examines the strengths and limitations of these methods, focusing on ObfNet, a neural network-based obfuscation algorithm, as a representative case study to contextualize our analysis. By analyzing ObfNet, we highlight its vulnerabilities and based on these findings we introduce LightNet and DenseNet as novel neural networks to identify ObfNet's limitations, particularly for larger and more complex data. We uncover challenges such as information leakage and explore the implications for maintaining privacy during remote neural network inference. Our findings underscore the challenges and possibilities to preserve privacy during remote neural network inference for UPDT algorithms, especially in resource-limited edge devices.

1 INTRODUCTION

In recent years, the rapid advancement of deep learning has significantly enhanced the complexity and capabilities of inference models, which now demand substantial computational resources. Deploying these models on Internet of Things (IoT) devices presents a formidable challenge due to their limited computational capacity and power constraints. IoT devices, such as smartphones, rely on battery power and need to conserve energy to function efficiently over extended periods, often requiring them to remain dormant and occasionally transmit data for processing. Given these constraints, offloading complex computations to remote servers or cloud-based backends, which can handle the heavy processing requirements of deep learning models, has become essential (Nieto et al., 2024). However, this approach introduces new concerns, particularly in terms of data privacy. Edge computing, which involves shifting computation and storage from centralized cloud servers to network edge nodes, offers notable advantages, including reduced end-to-end latency and minimized bandwidth usage. This paradigm is particularly beneficial when applying machine learning techniques to the vast amounts of data generated by IoT devices. However, even with these benefits, ensuring the privacy of data during remote processing remains a critical issue. IoT devices, due to their ubiquitous nature and the sensitive nature of the data they handle, are particularly vulnerable to privacy breaches during data transmission and processing. As a result, protecting the privacy of the data in these devices while maintaining the efficiency of machine learning processes is a key challenge (Zheng et al., 2019). Developing lightweight and efficient privacy-preserving methods that can operate effectively within these constraints is crucial. Such methods must not only safeguard sensitive information throughout the machine learning pipeline, from data collection to model inference, but also minimize communication and computational overhead to facilitate remote inference effectively. This is essential because high communication overhead not only causes delays and increased latency but also drives up data transfer costs, especially in IoT systems reliant on cellular networks where charges are based on data usage. Similarly, excessive computational overhead can drain the lim-

Yonan, Y., Abdullah, M. O., Nilsson, F., Fazeli, M., Patooghy, A. and Nowaczyk, S.
Privacy-Preserving Machine Learning in IoT: A Study of Data Obfuscation Methods.
DOI: 10.5220/0013458900003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 347-354
ISBN: 978-989-758-760-3; ISSN: 2184-7711

ited battery life of IoT devices. Efficient management of these overheads ensures real-time processing, scalability, and cost-effectiveness in IoT deployments (Zheng et al., 2019). One effective solution in this context is Utility-Preserving Data Transformation (UPDT) methods (Zheng et al., 2019)(Xu et al., 2020)(Feng and Narayanan, 2021)(Malekzadeh et al., 2020). UPDT techniques have emerged as an essential approach in privacy-preserving machine learning, particularly in the context of Internet of Things (IoT) and edge computing. These techniques aim to mask sensitive information within data while retaining its utility for tasks such as remote machine learning inference. UPDT methods are designed to be lightweight, suitable for resource-constrained IoT devices with limited computational and energy resources. UPDT methods must balance the dual objectives of preserving privacy and retaining data utility for inference. However, their effectiveness is often compromised by vulnerabilities, including residual data patterns that can expose sensitive information and scalability issues with complex datasets (Dhinakaran et al., 2024). This work seeks to address these challenges by evaluating the privacy-preserving capabilities and computational efficiency of UPDT methods, focusing specifically on their limitations and vulnerabilities. Using ObfNet (Zheng et al., 2019)(Xu et al., 2020) as a case study, we conduct a comprehensive analysis to uncover its shortcomings and explore the broader implications for UPDT methods. Our key contributions, a detailed evaluation of ObfNet as a case study, highlighting vulnerabilities in its ability to obfuscate sensitive data while maintaining computational efficiency and utility. These findings are then used as a basis for exploring and assessing other UPDT solutions. The introduction of LightNet and DenseNet, two novel architectures designed to stress-test UPDT methods on complex datasets. Generalized insights and recommendations for enhancing the robustness, scalability, and privacy guarantees of UPDT methods in resource-constrained environments.

2 BACKGROUND AND LITERATURE

Utility-Preserving Data Transformation methods are designed to transform data in a way that obscures, obfuscates, or eliminates sensitive information while preserving the attributes necessary for machine learning inference. These methods are especially relevant in privacy-preserving inference for IoT and edge computing, where devices need lightweight solutions that ensure privacy without compromising utility. An example of a UPDT method is anonymization, where sensitive identifiers such as names, addresses, or phone numbers are removed or replaced with pseudonyms. However, while anonymization can prevent direct identification, it often fails to protect against re-identification attacks, especially when combined with auxiliary datasets (Ano, 2017). Differential privacy (DP) on the other hand, provides a framework for quantifying the privacy guarantees of algorithms, ensuring that the removal or addition of a single data point does not significantly affect the outcome. This technique is particularly useful in scenarios where multiple queries are made to a database, as it provides robust privacy protection by adding noise to the query results. Although differential privacy provides strong privacy guarantees, it reduces data utility due to noise addition, especially in small or complex datasets. It also introduces implementation complexity (Abadi et al., 2016)(ha et al., 2019). In contrast, ObfNet(Zheng et al., 2019)(Xu et al., 2020) is a neural network-based UPDT algorithm designed for IoT applications. ObfNet transforms data at the edge device using the first, smaller half of a partitioned neural network with many-to-one mapping activation functions, elimiating sensitive attributes before transmitting them to the server for final inference, as described in Figure 1. According to the authors, many-to-one activation functions make data reconstruction virtually impossible, but recent research (Ding et al., 2022) suggests potential vulnerabilities in its obfuscation, as residual patterns may expose sensitive information. Even if only intermediate layers of a neural network are shared, these layers may still contain sensitive information. This is because the training methods used in the ObfNet algorithm (Zheng et al., 2019) do not explicitly differentiate between utility and sensitive data.

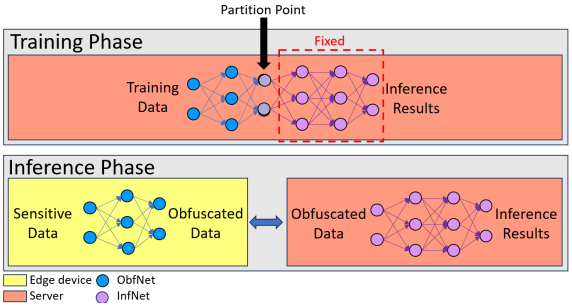


Figure 1: ObfNet training and inference phases with fixed InfNet and partitioned ObfNet.

Another example of a UPDT method is the Replacement AutoEncoder (RAE) (Malekzadeh et al., 2020), where an autoencoder is trained with raw time-series data as input and the same data as output, but

with sensitive information selectively replaced in specific time windows by neutral data. RAE have been combined with gradient reversal layer (GRL) (Ganin and Lempitsky, 2015) to protect demographic information (Feng and Narayanan, 2021). While ObfNet and RAE share similarities in their approaches to privacy-preserving transformation, they differ in the residual sensitive information they may leave behind. The authors of (Malekzadeh et al., 2020) designed training methods for RAE to selectively replace sensitive portions of the data while also validatating the removal of sensitive information through privacy loss metrics. These metrics -measured as the adversary's accuracy in inferring sensitive information- approach random guessing, indicating effective obfuscation. In contrast, ObfNet employs a many-to-one activation mapping without conducting comprehensive security evaluations to determine the extent of data removal. Consequently, it remains unclear whether ObfNet effectively removes sensitive information or if the transformed data may still be exploited for purposes beyond its intended utility. Privacy-preserving inference techniques focus on protecting sensitive data during the inference phase. A prominent approach is the use of a UPDT method like OfbNet(Zheng et al., 2019). In these networks, data is transformed or masked on the edge device before being sent to the server, where a subsequent, larger, model performs the final inference task. This approach aims to prevent the server from accurately reconstructing the original data, thus preserving privacy. Other approaches include CryptoNets, which adapt neural networks to process homomorphically encrypted data, enabling secure computation without decrypting sensitive information (Chen and Ran, 2019), and Generative Adversarial Models (GANs), which add controlled noise to data, creating obfuscation that can confuse adversaries during inference (Romanelli et al., 2019).

3 METHODOLOGY

This section details the security evaluation approach for examining ObfNet, with a focus on identifying potential vulnerabilities and risks of information leakage. These findings may also have implications for similar UPDT methods, shedding light on shared privacy challenges across this category. To examine ObfNet, we followed the guidelines outlined in the original ObfNet paper (Zheng et al., 2019) and used the source code found in (Xu et al., 2020) during the implementation of ObfNet. We used the CNN-based inference model, and both CNN and MLP-based ObfNets were explored, with bottleneck sizes

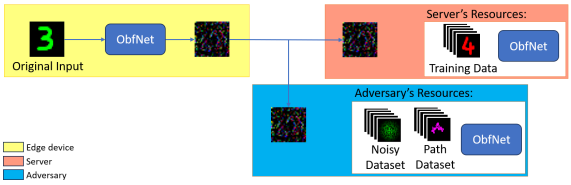


Figure 2: Attack scenario with server and adversary.

ranging from 8 neurons to 512 neurons. In our experiments with colored images, we tested bottleneck sizes from 8 to 1024 to examine the consequences of a larger bottleneck size as the authors claim it provides better obfuscation (Xu et al., 2020). When working with colored images, the size of the input layer of all architectures of ObfNets was modified from 28x28 neurons to 28x28x3 neurons to accommodate the additional color channels. The architectures of the ObfNets and the inference network can be found in (Xu et al., 2020). The attack scenario is as follows. We employ ObfNet to obfuscate MNIST images (Deng, 2012) as well as Colored-MNIST (a modified version of MNIST introduced later). Then the server attempts to reconstruct them or extract useful information, such as color, utilizing all available resources, including the public ObfNet network and the training dataset used to generate it. Another attacker in this scenario is an adversary intercepting the connection between the edge device and the server, assuming the obfuscated images are sent without encryption. The difference between the server and the adversary is that the latter lacks access to the original training data. The adversary also possesses background knowledge regarding the nature of the data being transmitted, such as recognizing that MNIST images depict numbers or that Colored-MNIST includes color information. The adversary also has the capability to gather or generate their own datasets. An overiew of both attackers and their respective resources are depicted in Figure 2. To quantify privacy, many methods tailored for obfuscation use similarity measurements between the original and obfuscated images, creating a similarity score (Raynal et al., 2020). However, in ObfNet's case, the obfuscated images look vastly different from the original ones, making a visual similarity measure a poor method for privacy quantification. Despite trying to estimate a full reconstruction of the original input from obfuscated images, many of our tests focus on extracting color information from obfuscated images since we know color is a feature irrelevant for digit classification and it should not remain in the obfuscated images. At the same time, the adversary is aware of the presence of color information in the obfuscated images. The attackers' ability to accurately predict the col-

Figure 3: Samples from the Colored-MNIST dataset.

Figure 4: Samples from the Noisy dataset.

ors of the obfuscated images serves as a metric for quantifying privacy. This feature-accuracy (F_a) represents the attackers' ability to extract private features from the obfuscated images and is calculated using $F_a = \frac{C}{T}$ where C is the number of correctly predicted colors, and T is the total number of obfuscated images. Given the seven equally distributed colors in our datasets (Explained later), random guessing would result in an expected F_a of **14.2%** for both the server and the adversary. This baseline accuracy provides a reference point. In practice, an ObfNet will reasonably be deemed compromised if this F_a exceeds twice the expected value, rounded up to **30%**. When it comes to datasets, in addition to using the original MNIST dataset (Deng, 2012), we generated our own datasets based on it. To avoid class imbalance, all colors are randomly selected an equal number of times. Colored-MNIST dataset is used for obfuscation and security assessment, meanwhile the adversary generates Noisy and Path datasets. Colored-MNIST is an extension of the original MNIST dataset. Initially, we select a random color from the predetermined set comprising {*red, green, blue, aqua, magenta, yellow, and white*}, which is subsequently applied to individual digit images from the MNIST dataset. Example images from the dataset can be seen in Figure 3. The motivation behind this dataset is to evaluate the color-prediction accuracy from obfuscated images, as the standard MNIST dataset lacks the color feature. Noisy dataset is generated by assigning each pixel a random intensity influenced by its distance from the center, creating a higher intensity towards the middle. The entire image is then assigned a random color from the Colored-MNIST palette. The dataset consists of 30 thousand training images. Example images can be seen in Figure 4. Path dataset aims to simulate a handwriting-like appearance by generating a random walk pattern. The process involves initiating the walk at the center and taking a step in a random direction for a predetermined number of steps. The entire image is subsequently assigned a random color from the set used in the Colored-MNIST dataset. Example images from the dataset can be seen in Figure 5. The

Figure 5: Samples from the Path dataset.

motivation for this dataset closely parallels that of the Noisy Dataset, where an adversary serves as the attacker. In this scenario, however, we assume that the adversary possesses slightly more background knowledge regarding the Colored-MNIST data. Specifically, we presume that the adversary is aware of the color information and the general shapes of the digits, resembling a random walk. Reconstruction network (RecNet) is an auto-encoder with an identical architecture to the MLP-based ObfNet found in (Xu et al., 2020), with a fixed bottleneck of 1024 neurons. The server uses RecNet to estimate the original input images based on obfuscated images. The server generates a unique RecNet for each ObfNet in an attempt to reverse its specific transformations. An overview of this attack is illustrated in Figure 6. Each ObfNet, whether MLP or CNN, is paired with its own unique RecNet counterpart. While all RecNets are identical in terms of architecture, the distinction lies in the data on which they are trained. The server obfuscates the entire training dataset using one ObfNet. RecNet is then trained using pairs of the original and obfuscated images to recreate the original images. This process is repeated for each ObfNet. Following the examples provided in (Xu et al., 2020), standard MNIST images are obfuscated using both MLP and CNN variants of ObfNet, with bottleneck sizes varying from 8 to 512 neurons. For each ObfNet, the server constructed a corresponding RecNet to estimate the original images. This identical attack is also tested using Colored-MNIST images to evaluate the impact of color on the reconstructed images. ColorNet adopts the architecture of the MLP-based inference network described in (Xu et al., 2020) with two slight modifications. The input layer of this network is adjusted from 28x28x1 to 28x28x3 to accommodate the color channels present in the Colored-MNIST dataset. The output layer is also adjusted from 10 to 7, reflecting the classification task among the seven available colors. The primary objective of this network is to target a specific feature that ObfNet aims to protect. This is achieved by classifying the obfuscated images based on their original digit colors. As color is not a necessary feature for digit classification, it is expected that ObfNet should have kept this feature private by removing it from the obfuscated images. The feature-accuracy of ColorNet serves as a determinant in assessing the security of an ObfNet. An overview of

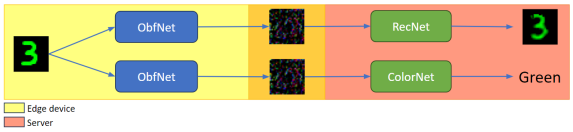


Figure 6: ColorNet and RecNet overview with example images.

this attack is illustrated in Figure 6. ColorNets are generated on the server and trained using obfuscated training images from Colored-MNIST. Each ObfNet is paired with its corresponding ColorNet. Both MLP and CNN variants of ObfNet are investigated, with bottleneck sizes varying from 8 to 1024 neurons.

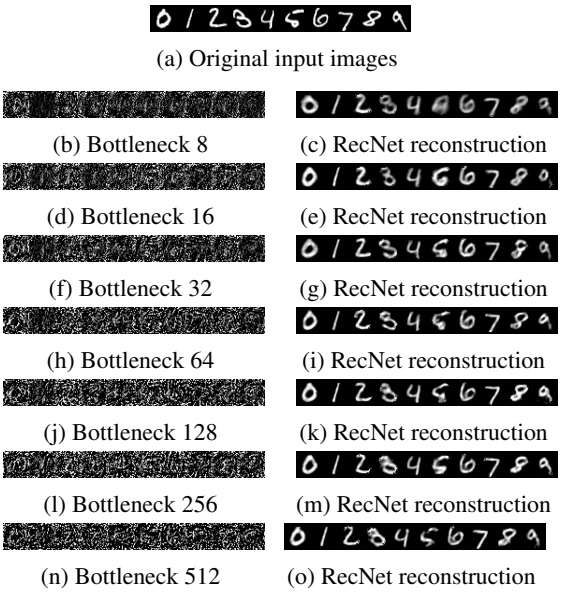
NoisyNet and PathNet are both ColorNets, but they are trained on obfuscated Noisy and Path datasets, respectively. The adversary utilizes these networks to learn the colors of the obfuscated Colored-MNIST images under the assumption that they do not have access to original training data, but only to a proxy. These networks are tested on CNN and MLP-based ObfNets with bottleneck sizes varying from 8 to 1024.

All of our attacks thus far rely on the fact that, as outlined in (Xu et al., 2020), the ObfNet network is public, thereby accessible to everyone. If ObfNet were trained and set up by a trusted third party, such that neither attacker had access to it, this approach would mitigate many risks associated with the network's security. In this attack scenario, both the server and the adversary only have access to the obfuscated images. However, the ability of the inference network at the server to classify the obfuscated images into digits suggests that the features of the digits are embedded within the obfuscated images. To further analyze the information embedded in the obfuscated images, both CNN and MLP-based ObfNets are employed to obfuscate the testing images in the Colored-MNIST dataset. This is done to see if the sensitive color information is still present in the images. The obfuscated images are processed through a t-SNE dimensionality reduction technique to reduce their initial 2352 dimensions into a more manageable 2-dimensional representation. After many tests, the perplexity value of the t-SNE hyperparameter was set to 15 for all subsequent t-SNE experiments. This choice balances preserving the global structure, such as the numbers' clusters, and capturing local structures, such as the colors' sub-clusters.

According to the work presented in (Zheng et al., 2019), ObfNet is a small-scale neural network designed to be deployed on resource-constrained edge devices. To validate this assertion, we utilized a pre-trained ResNet50 (He et al., 2015) as the inference network at the server and employed ObfNet to obfuscate ImageNette (Howard, 2019) images. ImageNette images are larger and more complex than MNIST, particularly in classification tasks. This poses a challenge to the extent to which ObfNet can maintain its lightweight and small-scale nature. When considering the proposed ObfNet architectures in the paper (Xu et al., 2020), simply reshaping the input to have the shape $(224, 224, 3)$ results in a model with excessive parameters. This, in turn, increases the size, the one-time communication overhead, and the RAM usage on the edge device. This significantly complicates the model training process and undermines the fundamental purpose of ObfNet as a small-scale neural network. For this reason, we introduce two new obfuscation networks, namely LightNet and DenseNet.

LightNet is a fully convolutional neural network (CNN) constructed for obfuscation of images in image classification tasks, specific for the ImageNette dataset (Howard, 2019). The CNN architecture consists of several critical layers designed to extract features from the input images and generate new obfuscated images that can be sent to the server. LightNet has no dense layers, making it very lightweight in terms of parameters, size, and communication overhead. LightNet uses ResNet50 (He et al., 2015) as an inference network and is trained following the same methodologies presented in (Xu et al., 2020) using ImageNette training images. LightNet processes pre-processed input images of size (224, 224, 3) using only convolutional layers. Max pooling reduces spatial dimensions, followed by dropout to prevent overfitting. Convolutional layers extract spatial features, with batch normalization stabilizing training. Gaussian noise is added as a regularizer. Finally, a transposed convolution upsamples the feature map back to (224, 224, 3).

DenseNet is a CNN with two dense layers as a bottleneck. This architecture is inspired by the two-dense layer design outlined in the original paper (Xu et al., 2020). DenseNet incorporates two down-sampling blocks comprising a convolutional layer followed by a maxpool layer. This design aims to reduce the dimensions of the image. The bottleneck size is fixed at 128 neurons to keep the number of parameters and size of this network to a minimum. DenseNet also uses ResNet as an inference network and is trained following the same guidelines in (Xu et al., 2020) using ImageNette training images. DenseNet processes pre-processed input images of size (224, 224, 3) using two convolutional layers, each followed by max pooling to progressively reduce spatial dimensions. A dropout layer helps prevent overfitting. A bottleneck dense layer (128 neurons) is used before a final dense



(a) Original input images

(b) Bottleneck 8 (c) RecNet reconstruction

(d) Bottleneck 16 (e) RecNet reconstruction

(f) Bottleneck 32 (g) RecNet reconstruction

(h) Bottleneck 64 (i) RecNet reconstruction

(j) Bottleneck 128 (k) RecNet reconstruction

(l) Bottleneck 256 (m) RecNet reconstruction

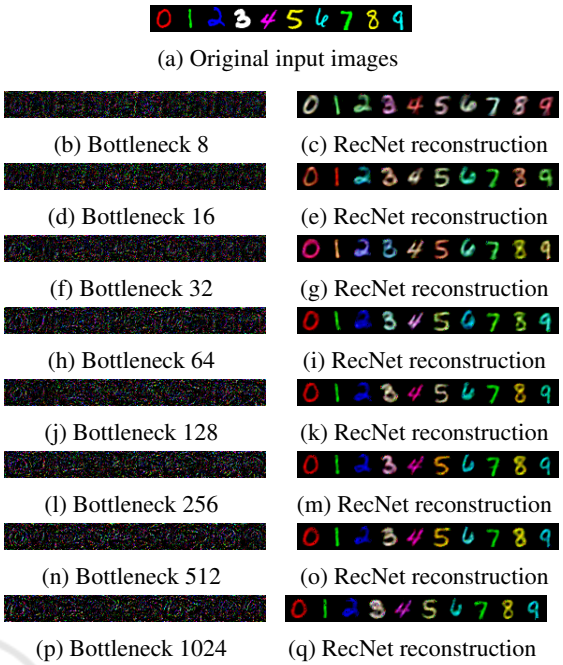
(n) Bottleneck 512 (o) RecNet reconstruction

Figure 7: MNIST reconstructions (MLP).

layer with $224 \times 224 \times 3$ neurons to reconstruct the output image. The final reshape operation restores the original image dimensions.

4 RESULTS

With a fixed bottleneck size of 1024 neurons, RecNet was deployed to estimate the original input to the ObfNet models obtained from the source code provided in (Xu et al., 2020). Our findings in Figure 7 align with those reported in the original paper, wherein the digit 'one' appears slightly darker than others when the bottleneck size is small. In the same Figure 7, the output of RecNet, or the reconstructed images, are displayed. The output of RecNet when estimating the original input of obfuscated Colored-MNIST images is illustrated in Figure 8 for MLP-based ObfNets. The server attempts to extract color information from the obfuscated Colored-MNIST images using ColorNet. The outcomes of these efforts are depicted in Figure 9. The feature-accuracy is showcased across the various bottleneck sizes in the plot. The adversary employs NoisyNet and PathNet to extract colors from the obfuscated images. The outcomes of these attempts are presented in Figure 10. These tests reveal that even the adversary can recognize the colors without access to the original training data with feature-accuracy exceeding 50%. The results thus far demonstrate that color and digit information are embedded within the obfuscated images. The obfuscated images underwent a t-



(a) Original input images

(b) Bottleneck 8 (c) RecNet reconstruction

(d) Bottleneck 16 (e) RecNet reconstruction

(f) Bottleneck 32 (g) RecNet reconstruction

(h) Bottleneck 64 (i) RecNet reconstruction

(j) Bottleneck 128 (k) RecNet reconstruction

(l) Bottleneck 256 (m) RecNet reconstruction

(n) Bottleneck 512 (o) RecNet reconstruction

(p) Bottleneck 1024 (q) RecNet reconstruction

Figure 8: Colored-MNIST reconstructions (MLP).

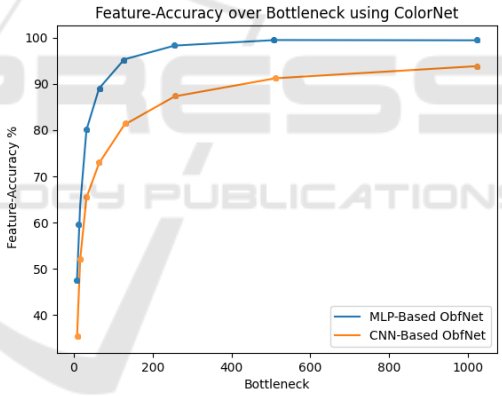
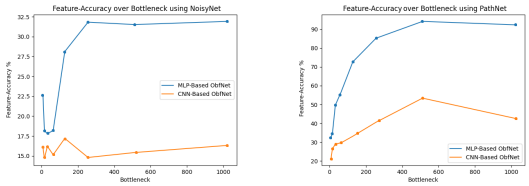


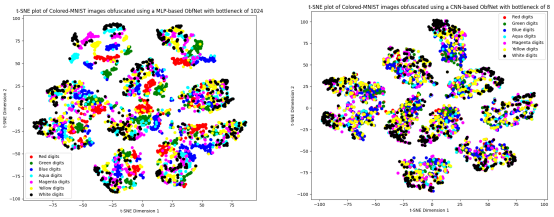
Figure 9: F_a of ColorNet with varying bottlenecks.

SNE dimensionality reduction procedure to analyze this phenomenon further. The results are visualized on a 2D graph in Figure 11 and they show ten primary clusters corresponding to the ten different digits. However, each cluster contains sub-clusters associated with colors, indicating that color information remains embedded in the obfuscated images. While this study focuses on ObfNet, the vulnerabilities we identified, such as information leakage during reconstruction attacks, are likely relevant to other UPDT methods. Similarities between ObfNet and techniques like Replacement AutoEncoders, and Anonymization suggest that residual sensitive information may persist when transformations are insufficiently selective to



(a) NoisyNet. (b) PathNet

Figure 10: F_a with varying bottlenecks.



(a) MLP, Bottleneck:1024 (b) CNN, Bottleneck:8

Figure 11: t-SNE of Colored-MNIST with two ObfNets.

protect specific sensitive information. This underscores a broader challenge within UPDT frameworks: achieving a balance between utility preservation and robust privacy. When dealing with larger and more complex data, scaling up ObfNet results in many trainable parameters and FLOPs. In Table 1, a comparison between ObfNets and our proposed models is presented. LightNet achieved 93% accuracy while DenseNet achieved only 59%. Obfuscated images for LightNet can be seen in Figure 12. DenseNet images are visually obfuscated.

Table 1: ObfNet vs. our models on 224x224x3 images.

Model	Parameters	FLOPs	Size
ResNet50	26.2M	7.73e9	105 MB
ObfNet-MLP	154.3M	3.08e8	617 MB
ObfNet-CNN	279.1M	6.46e8	1.11 GB
LightNet	20.7K	5.10e8	76 KB
DenseNet	25.0M	2.29e8	100 MB

5 CONCLUSION & FUTURE WORK

All the outcomes from the experiment mentioned above, particularly the t-SNE analysis, suggest that the ObfNet algorithm is more inclined to leak as much information as possible rather than removing information to protect privacy. Throughout all the conducted tests, the ObfNet algorithm consistently failed for various reasons each time. While the small bottleneck forces some information to be removed, the lack of mechanisms to control what is eliminated underscores a broader challenge not just for ObfNet but for similar UPDT methods. This highlights the need for more advanced approaches that explicitly prioritize the removal of sensitive information while maintaining utility. The results of RecNet on obfuscated test images from the Colored-MNIST dataset can be seen in Figure 8 for the MLP variant. The CNN variant has very similar results. RecNet not only approximated the original input shape and digit but also accurately inferred the digit color from the obfuscated image, given that the bottleneck of the ObfNet is large enough. The obfuscated MNIST examples presented in (Xu et al., 2020) were fully reconstructed in Figure 7, effectively undermining the examples obtained in the source code. This further substantiates the vulnerabilities of ObfNet. This observation reinforces the broader challenge faced by UPDT methods, as similar reconstruction risks may arise when sensitive attributes are insufficiently obfuscated. All ColorNet models display the substantial color information embedded within the obfuscated images, achieving peak feature-accuracies of 99.46% and 99.40% on MLP-based ObfNets with bottleneck sizes 512 and 1024, respectively (Figure 9, Figure 10). The results of the t-SNE graphs further strengthen the findings of ColorNet in Figure 11, with ten primary clusters in the MLP-based obfuscated images, each of these clusters corresponds to one of the ten digits. However, each main cluster displays distinct sub-clusters corresponding to colors. This suggests that color information remains embedded within the obfuscated images and is susceptible to exploitation. This highlights a broader challenge for UPDT methods: the residual data left behind after the transformation often retains sensitive attributes, such as color information, which can compromise privacy. Ensuring that transformed data is free from exploitable patterns while preserving its utility remains a critical hurdle for these methods. LightNet failed to obfuscate images effectively, as demonstrated by Figure 12. This failure emphasizes the need for better training methods where feature removal is controlled. DenseNet, incorporating dense layers and a small bottleneck, managed to obfuscate images but at the cost of significantly reduced classification accuracy due to the complexity of the ImageNette dataset. The balance between bottleneck size, feature removal, and task complexity remains challenging to achieve with current training methods. The

Figure 12: LightNet output on ImageNette (obfuscated).

images were preprocessed using ResNet's preprocessing function (TensorFlow,). Therefore, any differences observed in Figure 12 can be attributed to the preprocessing function. The original paper of ObfNet (Xu et al., 2020) states that ObfNet is lightweight and can run on devices without acceleration for inference. This claim holds primarily for MNIST and other small-scale, simple datasets. However, this feasibility diminishes as dataset size and complexity increase, leading to a substantial rise in network parameters, size, and FLOPs. This observation is not unique to ObfNet but reflects a common challenge for many UPDT methods when scaling to more complex data: the trade-off between computational efficiency and robust privacy preservation as the size of the data that needs to be transformed grows. The original paper (Xu et al., 2020) also claims that "when more neurons are used in the first hidden layer of O_M, the overall darkness levels of the obfuscation results of all digits are equalized, suggesting a better obfuscation quality"-however, our test results in Figure 7 show the opposite. As the bottleneck size increases, ObfNet inadvertently retain more information, making sensitive data more susceptible to leakage. This reliance on visual indicators of obfuscation, rather than robust privacy metrics, is a broader issue across UPDT techniques. Each dataset comes with its unique privacy requirements and characteristics, making it difficult to establish a universal privacy metric that applies to all cases. Furthermore, the lack of well-defined design principles in UPDT methods is a common challenge as each dataset is different (Malekzadeh et al., 2020). For example, the results from LightNet (Figure 12) demonstrate that without explicit mechanisms to enforce effective obfuscation, networks trained to prioritize utility-such as inference accuracy-may inadvertently leave sensitive data insufficiently transformed. This issue is further exacerbated by training methodologies that do not impose strong constraints for selective feature removal, which can result in residual sensitive information remaining within the transformed datasets. Addressing these shortcomings is essential for improving the scalability, robustness, and privacy guarantees of UPDT architectures.

REFERENCES

(2017). Geo. l. tech. rev. 202.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. pages 308–318.

Chen, J. and Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dhinakaran, D., Sankar, S. M. U., Selvaraj, D., and Raja, S. E. (2024). Privacy-preserving data in iot-based cloud systems: A comprehensive survey with ai integration.

Ding, X., Fang, H., Zhang, Z., Choo, K.-K. R., and Jin, H. (2022). Privacy-preserving feature extraction via adversarial training. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1967–1979.

Feng, T. and Narayanan, S. (2021). Privacy and utility preserving data transformation for speech emotion recognition. In *2021 9th International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 1–7.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation.

ha, T., Dang, T., Dang, T. T., Truong, T., and Nguyen, M. (2019). Differential privacy in deep learning: An overview. pages 97–102.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Howard, J. (2019). Imagenette. https://github.com/fastai/imagenette.

Malekzadeh, M., Clegg, R. G., Cavallaro, A., and Haddadi, H. (2020). Privacy and utility preserving sensor-data transformations. *Pervasive and Mobile Computing*, 63:101132.

Nieto, G., de la Iglesia, I., Lopez-Novoa, U., and Perfecto, C. (2024). Deep reinforcement learning techniques for dynamic task offloading in the 5g edge-cloud continuum. *Journal of Cloud Computing*, 13(1):94.

Raynal, M., Achanta, R., and Humbert, M. (2020). Image obfuscation for privacy-preserving machine learning.

Romanelli, M., Palamidessi, C., and Chatzikokolakis, K. (2019). Generating optimal privacy-protection mechanisms via machine learning. *CoRR*, abs/1904.01059.

TensorFlow. Preprocesses a tensor or numpy array encoding a batch of images. https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet/preprocess_input.

Xu, D., Zheng, M., Jiang, L., Gu, C., Tan, R., and Cheng, P. (2020). Lightweight and unobtrusive data obfuscation at iot edge for remote inference. *IEEE Internet of Things Journal*, 7(10):9540–9551.

Zheng, M., Xu, D., Jiang, L., Gu, C., Tan, R., and Cheng, P. (2019). Challenges of privacy-preserving machine learning in iot. In *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, SenSys '19. ACM.