Improving Robustness of Deep Learning Models and Privacy-Preserving Image Denoising

Hadi Zanddizari
University of South Florida

Improving Robustness of Deep Learning Models and Privacy-Preserving Image Denoising

by

Hadi Zanddizari

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Date of Approval:
March 22, 2022

Keywords: Black-box attack, internet of things, cloud computing, image denoising, sparse coding

Dedication

This work is completely dedicated to my respectful parents without whose constant support this thesis paper was not possible.

Acknowledgments

First, I would like to express my sincerest gratitude to my advisor and my lifelong friend, Professor J. Morris Chang, for the priceless guidance of me through my Ph.D. journey over the last four years. I want to say thank you to Prof. Chang for unconditionally supporting me with academic knowledge, research instructions, professional career advice, and particularly, guiding me to choose research topics that are among the most interesting ones in the industry and academia.

I would also like to thank the rest of the professors on my dissertation committee: Dr. Nasir Ghani, Dr. Xinming Ou, Dr. Lu Lu, and Dr. Ismail Uysal. The knowledge that I have gained from them during coursework or discussions helped enrich my dissertation.

Last but not the least, I would like to express my deepest gratitude to my family. This dissertation would not have been possible without their warm love, continued patience, and endless support.

Table of Contents

List of Tables

List of Figures

Abstract

Applications of deep learning models and convolutional neural networks have been rapidly increased. Although state-of-the-art CNNs provide high accuracy in many applications, recent investigations show that such networks are highly vulnerable to adversarial attacks. The black-box adversarial attack is one type of attack that the attacker does not have any knowledge about the model or the training dataset, but it has some input data set and their labels.

In this chapter, we propose a novel approach to generate a black-box attack in a sparse domain, whereas the most critical information of an image can be observed. Our investigation shows that large sparse (LaS) components play a crucial role in the performance of image classifiers. Under this presumption, to generate an adversarial example, we transfer an image into a sparse domain and add noise to the LaS components. We propose a comprehensive evaluation and analysis to support our idea in chapter one.

In chapter two, we propose a new preprocessing approach that can enhance the robustness of skin lesion classification. Machine learning models based on convolutional neural networks have been widely used for automatic recognition of lesion diseases with high accuracy compared to conventional machine learning methods. In this research, we proposed a new preprocessing technique to extract the skin lesion dataset's region of interest (RoI).

We compare the performance of the most state-of-the-art convolutional neural networks classifiers with two datasets that contain (1) raw and (2) RoI extracted images. Our experiment results show that training CNN models by RoI extracted dataset can improve the prediction accuracy. It significantly decreases the evaluation and training time of the classification task.

Finally, we propose a secure and robust image denoising approach. Image denoising aims to obtain the original image from its noisy measurements. While the quality of image denoising has been increasing over the years, the complexity and the required memory to implement the denoising task have also been increased accordingly. With such advancements and the unlimited computing resources available in the cloud, trends to transfer the image denoising task to the cloud have grown over the past years. However, it is still quite challenging to utilize cloud-based resources without compromising users' data privacy while maintaining the quality of image denoising. In this chapter, we propose a novel lossless privacy-preserving image denoising approach that protects the users' privacy and simultaneously keeps the quality of the denoising task.

Our proposed approach is suitable for computationally constrained devices such as many IoT devices. In this method, we use two random keys to permute and perturb the noisy image patches. The cloud service provider implements the denoising task on the encrypted signal. After denoising, the output signal is still encrypted, and the real user who has access to the keys would be able to decrypt the denoised image. We evaluate the security of this method against known-plaintext, brute-force, and side-channel attacks. In addition, we theoretically prove the lossless property of this method. To verify the applicability of this approach, we implemented our experiments on multiple real images, and two well-known evaluation metrics were used to compare our results with the baseline.

Chapter 1: The Robustness of Deep Learning Models against Adversarial Attacks

Applications of machine learning (ML) models and convolutional neural networks (CNNs) have been rapidly increased. Although state-of-the-art CNNs provide high accuracy in many applications, recent investigations show that such networks are highly vulnerable to adversarial attacks. The black-box adversarial attack is one type of attack in which the attacker does not have any knowledge about the model or the training dataset, but it has some input data set and their labels. In this chapter, we propose a novel approach to generate a black-box attack in a sparse domain, whereas the most important information of an image can be observed [1].

Our investigation shows that large sparse (LaS) components play a critical role in the performance of image classifiers. Under this presumption, to generate an adversarial example, we transfer an image into a sparse domain and put a threshold to choose only k LaS components. In contrast to the very recent works that randomly perturb k low frequency (LoF) components, we perturb k LaS components either randomly (query-based) or in the direction of the most correlated sparse signal from a different class. We show that LaS components contain some middle or higher frequency components information, which leads to fooling image classifiers with fewer queries.

We demonstrate the effectiveness of this approach by fooling six state-of-the-art image classifiers, the TensorFlow Lite (TFLite) model of Google Cloud Vision platform, and the YOLOv5 model as an object detection algorithm. Mean squared error (MSE), and peak

[1]This chapter was published in IEEE Transactions on Emerging Topics in Computational Intelligence [1]. Permission is included in Appendix A.

signal-to-noise ratio (PSNR) are used as quality metrics. We also present theoretical proof to connect these metrics to the perturbation level in the sparse domain.

1.1 Introduction

By the ever-increasing demands for analyzing and processing large datasets, ML algorithms and particularly deep learning techniques have become the center of attention of many companies and service providers. The remarkable performance of CNNs for image segmentation, classification, and object tracking could provide acceptable solutions for many problems encountered in computer vision and biomedical engineering [2, 3, 4]. While almost CNNs perform well and provide high accuracy, their robustness toward some malicious attacks still is not acceptable [5, 6, 7]. Applying some perturbation on the input data may undermine the high accuracy of a classifier since ML models are usually trained and deployed in benign settings. In other words, they do not consider certain scenarios in which an attacker can compromise the performance of the system.

Recently, many works have been proposed to point out the vulnerability of CNNs against adversarial scenarios [8, 9, 10, 11, 12]. By slightly perturbing the input data, the ML classifier may fool and predict a wrong label. If this perturbation is small enough to the human eyes, then the perturbed image is called an adversarial example [6, 13, 14]. This problem can be viewed from a different perspective; if we add a limited perturbation to an image, while human eyes may detect the perturbation, still we expect the classifiers to classify correctly. It opens up a new horizon of the robustness of ML models against adversarial examples.

An adversarial example can be obtained by solving the following minimization problem

$$||r||_2 \quad \text{s.t.} \quad C(x+r) \neq C(x) \tag{1.1}$$

where r is adversarial perturbation, $||.||_2$ is the Euclidean norm or ℓ_2 norm, x is the legitimate image (original image), and $C(.)$ yields the classifier's output label. Based on (1.1), there are two factors in generating adversarial examples, first having a minimum perturbation on the legitimate image, and the second, fooling the classifier output.

Misclassification and targeted misclassification attacks are two major goals of adversarial examples. In the misclassification attack, an adversary tries to fool the ML classifier by misclassifying a legitimate example to different classes other than the original one. For example, a legitimate image with label 1 of the MNIST (Modified National Institute of Standards and Technology) dataset is perturbed in such a way that ML classifier yields an output label belongs to {0,2,3,4,5,6,7,8,9}, yet not 1. The attacker tries to fool the classifier to yield a targeted label in targeted misclassification. For example, the same legitimate image with a label 1 is labeled as a specific number like 8 by the classifier. In this chapter, we focus on misclassification attacks.

Adversarial examples can be generated based on two different approaches: white-box and black-box. In white-box attacks, the attacker has comprehensive knowledge about the training dataset, model parameters, number of CNN layers, loss function, and the whole structure of the model. There are numerous works based on white-box attacks, such as fast gradient sign method (FGSM) [15], beyond the image space approach that uses physical space features of 3D images, [16], deepfool [17], Jacobean-based Saliency Map Attack (JSMA)[18]. For example, FGSM generates an adversarial perturbation for a given legitimate image by computing the gradient of the cost function with respect to the legitimate image of the ML algorithm as follows:

$$x^{*} = x + \epsilon \; sign\left(\nabla_{x}\mathcal{J}\left(x, c\right)\right) \tag{1.2}$$

where ϵ denotes a small scalar value which regulates the perturbation's level, c is the input label, $\mathcal{J}()$ denotes the model cost function, ∇_x is the gradient of the trained model with respect to the legitimate image, and $sign(.)$ is the common mathematical function which yields the sign of its input argument. The common property of white-box attacks is utilizing the model's information for generating the adversarial example. In contrast, the black-box attack does not have any information about the model's structure, parameters, and training dataset[19, 20, 21, 22]. This type of attack is more practical because access to the training dataset is not possible in many cases. Also, some information such as the model's parameters, number of layers, and loss function may not be public.

Black-box attacks can be separated into three categories: non-adaptive, adaptive, and strictly black-box attacks [13]. In a non-adaptive black-box attack, an attacker can have access only to the distribution of the training dataset [23]. In the adaptive black-box case, the attacker does not have any information about the distribution of the dataset; however, she can access the target model as an oracle. It means the attacker can query the output labels of legitimate samples as well as adversarial samples [24, 25]. In the strict black-box attack, the attacker does not have access to the dataset's training distribution, and she cannot adaptively modify the input query to observe the model's output. In other words, an attacker can query the legitimate input samples, but if she slightly perturbs an input sample to observe its output label, the system identifies this process as a malicious attack [26, 13]. Although these types of systems may provide a high level of security, in many real cases input samples may be very similar to each other and as a result, there is no need to block the user. Adaptive black-box attacks are more applicable than non-adaptive or strict black-box attacks as they do not have any knowledge about the distribution of the training dataset and assume the system would not block a user by evaluating a limited number of close queries. However, if the number of queries increases, the system may detect a probable malicious attack.

In [27], authors proposed generating adversarial examples based on perturbing one pixel of an image through differential evolution. Although this method could fool almost all CNN models due to the inherent features of differential evolution, there is no limit to the number of queries to attack the model. Papernot et al. [19] proposed a practical approach for generating adversarial examples based on Jacobian-based dataset augmentation technique to obtain new synthetic training samples. After having an adequate number of samples and corresponding labels, they train a local model and apply a white-box attack (such as FGSM) on this locally trained model to generate adversarial examples. They use the transferability property of ML algorithms [20]. Transferability is a property that enables us to apply adversarial examples generated by a model on another model with the same or different architecture. The applicability of such attacks mainly revolves around the transferability property of ML models and having enough large datasets for training the local model. Recently, Hosseini et al. [28] proposed a three-step null labeling method to block the transferability property of the ML models. In the first step, they train the model based on clean data, then add some perturbations to the input data, and based on some threshold and probability functions, and they assign the label 'Null' to the perturbed image. Then, they retrain the model with clean and new adversarial examples with null labels. This approach enables the model to detect the input adversarial examples by predicting as a 'Null'. The previous black-box attacks try to generate adversarial examples based on a white-box approach. In other words, they train a local fake model, then apply a white-box attack to generate adversarial examples.

Some black-box approaches are not based on the white-box approaches. In [24], the effectiveness of restricting the search for adversarial images to a low-frequency domain has been investigated. After focusing on the lower frequency subspace, they randomly perturb the components while restricting the perturbation level. It can be described as adding a low-filtered random noise to the legitimate image. This approach could outperform many black-box attacks. Y. Sharma et al. [25] used discrete cosine transform (DCT) dictionary to map the image into the frequency domain. Then they put a hard threshold for choosing LoF components. After transformation into the frequency domain, most of the frequency components have small values, and only a few have large values. This property of the frequency domain is well known as a sparse representation of an image. Then, by applying perturbations on the LoF components, they could generate faster and more transferable adversarial examples. This approach can completely bypass most of the top-placing defense strategies at the NeurIPS 2017 competition. The authors also investigated the effect of perturbation on high frequency (HiF) components, but their results show that LoF components are the ones that mostly affect CNN models. We were motivated by the aforementioned work and used DCT dictionary to transfer images into the sparse (frequency) domain. Then, instead of putting a hard threshold for choosing only k LoF components, we selected k LaS components where some low, middle, and high-frequency components are picked up. In section 1.2.1, we show the difference between LaS and LoF components.

Focusing on LaS components has been used in many image processing and compression techniques. The JPEG codec [29] takes advantage of this property in order to compress the images. Because the most critical features and information of an image are available in the LaS components and not just LoF components [29]. Intuitively, image classifiers mostly consider specific components that bear more image information. We verify this property of image classifiers by implementing systematic experiments (section 1.2.2). We propose adding noise to LaS components in two scenarios. In the first scenario, we randomly perturb LaS components, and by restricting the perturbation level, the number of required queries to fool the state-of-the-art classifiers is evaluated. Our experiment results show that the proposed approach can fool the classifiers with fewer queries compared to the very recent approach, which works based on LoF components [25]. In the second scenario, a directed attack, we suppose a few images from each class are available. Given a legitimate image, we perturb its LaS components in the direction of the most correlated sparse sample from a different class. Our experiments show that this method can successfully fool the state-of-the-art CNN classifiers.

In this chapter, the summary of our contributions are as follows:

- We introduce a black-box approach to generate adversarial examples in the sparse domain to fool the ML algorithms such as CNN models, support vector machine (SVM) classifiers, object detection algorithm (YOLOv5), and model trained by the Google Cloud Vision API.

- In contrast to the recent black-box attacks that focused on LoF components, we show that the LaS components can fool the classifiers with fewer queries.

- We proposed an analytical approach to show the relationship between the perturbation level in the sparse domain and its effect on the pixel domain. Our results show that the proposed method decreases the number of required queries to fool the ML models and increases the misclassification rate of ML models.

1.2 Sparsity

Sparsity has been widely used in many applications such as image denoising, deblurring, super-resolution, and compression [30, 31]. An image signal $X \in \mathbb{R}^{p\times q}$ can be reshaped to a vector $x \in \mathbb{R}^{N=p\times q}$ where N is the number of pixels. Dictionary $D \in \mathbb{R}^{N\times L}$ is a matrix which linear combination of its columns d_i can approximately represent the x as follows:

$$x = \sum_{i\in\{1,2,..L\}} s_i d_i = Ds \tag{1.3}$$

where $s \in \mathbb{R}^L$ is the weight vector. If D provides a weight vector with only k large and $l-k$ negligible or zero elements, then D and s can be called as a sparsifying dictionary and sparse representation of input x, respectively. For brevity, by the rest of this work, we omit the 'sparsifying' and refer to the dictionary as a sparsifying dictionary. There are some fixed dictionaries based on analytical approaches such as Fourier or wavelet transform which can be designed very fast. In this work, we used DCT dictionary which is an orthonormal matrix $(D \in \mathbb{R}^{N\times N}\ and\ ||d_i||_2 = 1)$. The coefficients of DCT dictionary can be obtained as follows:

$$\begin{aligned} d_{i,j} &= a_{i,j}\cos\frac{\pi(2i-1)(j-1)}{2N} \qquad i,j \in 1,2,\dots,N \\ a_{i,j} &= \begin{cases} \sqrt{\frac{1}{N}} & j=1 \\ \sqrt{\frac{2}{N}} & j\neq 1 \end{cases} \end{aligned} \tag{1.4}$$

where $d_{i,j}$ corresponds to the entry of ith row and jth column of DCT dictionary. If we transfer an image into the DCT domain, zeroing small components will have negligible effects on the visual information of the image. For example, Fig. 1.1 illustrates this property. The original image was transferred into the sparse domain via DCT dictionary and forced 70%, 80%, and 90% of its small components to zero then transformed back into the pixel domain. Reconstructed images based on only 30%, 20%, or 10% of its LaS components can still preserve lots of visual information of the image.

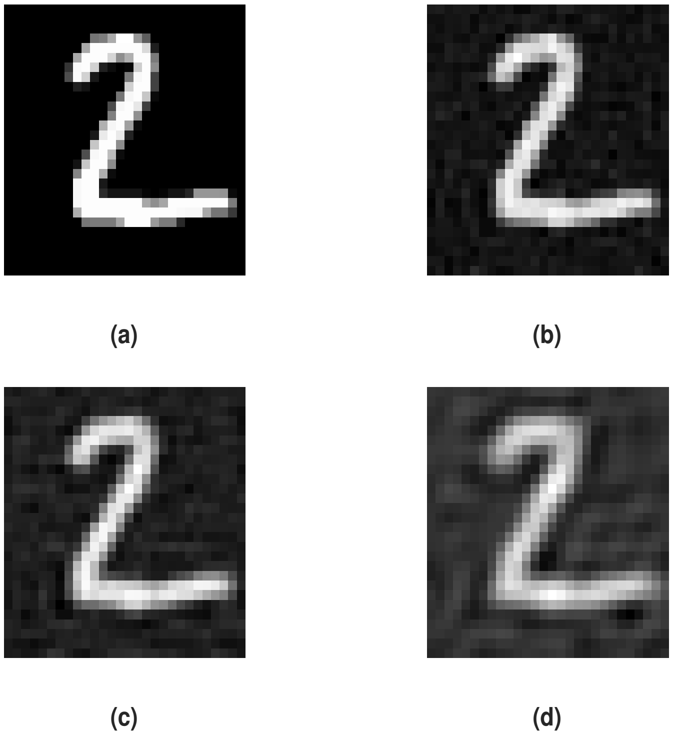


Figure 1.1: Transferring image into the sparse domain and zeroing small elements of sparse signal: (a) original image, (b) zeroing 70%, (c) 80%, and (d) 90% of small elements.

1.2.1 Difference Between LaS and LoF Components

The sparse domain enables us to access the important frequency components of an image. Components may belong to low, middle, or high-frequency bands. Regardless of the frequency bands, if we choose some top-ranked components, those specific components can

Table 1.1: The effect of keeping only 50% or 30% of LaS, LoF, and HiF components on the accuracy of six CNN models(%).

Model	Ground Truth Accuracy (All components)	50% of LaS	30% of LaS	50% of LoF [24] & [25]	30% of LoF [24] & [25]	50% of HiF [25]	30% of HiF [25]
MobileNets	90.72	89.14	83.75	77.14	76.27	29.79	15.71
ResNet50	91.37	90.73	87.59	79.30	73.29	20.89	16.13
DenseNet121	92.29	91.27	88.05	79.76	77.84	26.31	16.74
InceptionV3	93.27	92.6	90.32	80.83	79.40	31.42	25.93
Efficient-B0	94.30	93.83	90.59	79.07	70.57	36.54	27.16
Efficient-B1	95.46	94.78	91.06	80.25	75.85	37.36	29.47

belong to any frequency band. Some images may have some information in the middle or even higher frequencies; as a result, they would have LaS components corresponding to the middle or higher frequencies.

To evaluate the intersection level between LaS and LoF components, we used 10000 color images of size 256x256 pixels. The images had three color channels, and we mapped each channel into the sparse domain separately. Then we selected N = k× k× 3 LaS and LoF components. For chosen k = 8, k = 16, and k = 32, the number of components are N = 192, N = 768, N = 3072, respectively. Figure 1.2 shows how many non-intersecting components are available between LaS and LoF components. For k = 8, the mean of non-intersecting components is 77, i.e., more than 40% of the LaS components belong to the middle or higher frequencies components. For k = 16 and k = 32 the mean of non-intersecting components are 229 and 983, i.e., 39% and 32% of the LaS components do not belong to the low-frequency space. This experiment shows that the LaS components do not completely overlap with the LoF components, and some critical information of the image signals may belong to the middle or high-frequency bands. In other words, for every image, different bands have different information; as a result, we cannot limit the critical information of an image to only its low-frequency space. In the next section, we evaluate the effects of manipulating different frequency bands on the performance of CNN models.

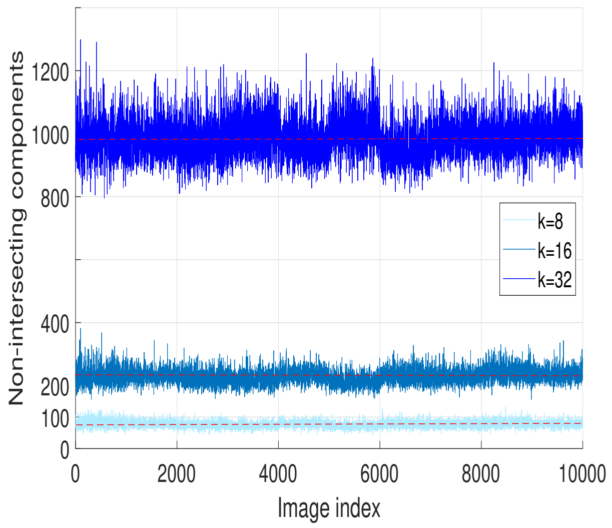


Figure 1.2: The number of non-intersecting components of each image.

1.2.2 Effect of LaS Components on CNN Models

Sparse transformation enables us to compact the signal's energy into a few components. On the other hand, many image classifiers work based on pixel domain and they do not directly consider the sparse domain. A question that may arise here is: "how much can manipulating LaS, LoF, or HiF components affect classifiers' performance?". This chapter empirically shows that the LaS components are the most important part of images that affect the classifiers' performance. Our experiment was implemented over six state-of-the-art CNN models namely, EfficientNet-B0 and B1 [32], ResNet50 [33], InceptionV3 [34], MobileNets [35], and DenseNet121 [36].

We used the CIFAR-10 dataset, a color and balanced image dataset with a complex background. This dataset contains 50000 training samples and 10000 test samples belonging to 10 classes. We trained these models with 50000 training samples, and then we input the original 10000 test samples (without any changes or manipulation) to obtain the ground truth accuracy of each trained model (Table I). In the next step, via DCT dictionary, we transferred all 10000 test samples into the sparse domain. Then we kept 50% and 30% of LaS, LoF, and HiF components and zeroed the rest of the components. We transformed each image back to the pixel domain and input them into the same trained model. To further clarify, after putting these thresholds, we obtained 6 test datasets, two for Las components, two for LoF components, and two for HiF components.

As shown in Table 1.1, the accuracies belonging to LaS components test datasets are much closer to their corresponding ground truth accuracies. While keeping only LoF or HiF components leads to a considerable loss of accuracy. It shows that if we only focus on LoF or HiF components; we lose some components that affect the decision boundaries of CNN models. For example, Efficient-B1, one of the best image classifiers introduced by Google in 2019, has an accuracy of 95.46% for the original test dataset. If we keep only 50% of LaS components, the accuracy is almost the same 94.78%. If we keep 50% of LoF and HiF components, the accuracies are 80.25% and 37.36%, respectively. To elucidate, only 50% of LaS components affect classifiers, and the other 50% components do not affect the accuracy much.

This experiment helps us determine which frequency components mainly affect the CNN models. By having this information, we would be able to add perturbation on important components to fool image classifiers. Also, this experiment verified the results of [25] that showed the importance of LoF vs. HiF components. They concluded that perturbing LoF components is more effective than perturbing HiF components. For brevity, we omitted the results of our experiments over other CNN models and different threshold levels, which had the same results to verify our assumption. We release our code publicly for reproducibility. In the next section, we add a limited perturbation to LaS and LoF components to see which of them can fool the classifiers in fewer queries.

1.3 Perturbing LaS Components

There is no prior information about the model's parameters and distribution of the training dataset in the adaptive black-box attack, yet the attacker can query the label of legitimate sample and corresponding perturbed sample. However, if the number of queries increases, the system may identify malicious activity. An adversarial attack is more practical if it fools classifiers in fewer queries. We designed a systematic experiment to evaluate the effectiveness of adding perturbation on LaS components. Our results demonstrate that the proposed approach requires fewer queries to fool image classifiers. In this experiment, six CNN models (EfficientNet-B0 and B1, ResNet50, InceptionV3, MobileNets, DenseNet121) were used. We trained all models with 50000 training samples of the CIFAR-10 dataset. We used 10000 test samples of the CIFAR-10 dataset that had never been used in the training process to apply the attacks. We utilized the DCT dictionary to transfer test samples into the frequency domain. We used a Gaussian noise with zero mean and variance 1 to generate noise, and to have a fair comparison with [25], we defined the MSE less than 0.001 as a successful attack. We compared adding noise to k = 8 LaS and LoF components. In Fig. 3.6, the histograms of the required number of queries to successfully fool the aforementioned CNN models are demonstrated. The distributions of successful attacks show that manipulating LaS components can fool the CNN models in fewer queries. Figure 1.4 shows the number of all misclassified images in query less or equal to 10. This experiment first evaluated the models' prediction for each legitimate sample. If a model predicted a legitimate sample wrongly, we put aside that sample and did not involve it in the experiment (because it was already misclassified). Hence, the number of misclassified images in Fig. 3.6 and 1.4 are only due to the perturbation on samples.

1.4 Case Study: Directed Perturbation

This section proposes a method for adding noise to the LaS components to fool the model into a specific direction. In the black-box approach, the attacker can use some samples that have never been used for the training stage. Then, the attacker can verify or find the input sample's label by observing the output of the objective model. This section assumes the attacker can have multiple samples of each class and its labels. Suppose the available dataset is $X = \{x_i\}_{i=1}^{i=p}$ which contains p samples and each sample belongs to one class out of

m available classes, i.e., $C(x_i) \in \{c_j\}_{j=1}^{j=m}$. We map all samples of the dataset into the sparse domain via DCT dictionary D. Doing so, $S = \{s_i\}_{i=1}^{i=p}$ would be obtained where s_i is the sparse representation of the x_i. In the sparse domain, we keep the k LaS components and force the rest of the components to zero. Then each sparse vector is normalized. Doing so, we would have

$$\hat{S} = \{\hat{s}_i\}_{i=1}^{i=p}, ||\hat{s}_i||_0 = k, ||\hat{s}_i||_2 = 1 \tag{1.5}$$

where $||.||_0$ is the zero-norm of a vector which counts the number of non-zero elements of a vector. Sparse vector $\hat{s}_i$ contains information of the positions and normalized values of the k largest elements of s_i which belong to class $C(s_i)$. Then for a given $(\hat{s}_i,\ C(s_i))$, we find the most correlated sparse vector $(\hat{s}_j,\ C(s_j) \neq C(s_i))$. In other words, sparse vector $\hat{s}_j$ is the closest sparse vector to the $\hat{s}_i$, but they belong to different classes. We used the inner product of two vectors $\langle \hat{s}_i, \hat{s}_j \rangle$ to calculate the correlation. If we change the k most important elements of $\hat{s}_i$ with respect to the k most important elements of $\hat{s}_j$, some information and features of $\hat{s}_j$ can be transferred into the $\hat{s}_i$. If some nonzero elements of $\hat{s}_i$ and $\hat{s}_j$ have the same positions and close values, there is no need to change or manipulate them. Because they have common information and changing them cannot help for fooling classifier and may bring unnecessary perturbation in the pixel domain. To prevent this probable issue, we subtract these two vectors to obtain the difference d_{ij} as follows:

$$d_{ij} = \hat{s}_i - \hat{s}_j \tag{1.6}$$

Then, we subtract a multiplier of d_{ij} from the original sparse vector s_i to obtain sparse adversarial example $\widetilde{s}_i$ as follows:

$$\widetilde{s}_i = s_i - \delta d_{ij} \tag{1.7}$$

where δ is a scalar number that controls the level of directed perturbation. Then, we transfer back the adversarial sparse vector $\widetilde{s}_i$ to the pixel domain via dictionary D as follows:

$$\widetilde{x}_i = D\widetilde{s}_i \tag{1.8}$$

where $\widetilde{x}_i$ is the adversarial example. Since the response of ML classifier for s_j is $C(s_j)$, when we add the elements of $\hat{s}_j$ to the $\hat{s}_i$, the classifier may be fooled. By choosing δ and k properly, ML classifiers can be fooled. Two scalar parameters k and δ control the level of perturbation. When we increase these scalars, the level of perturbation in the pixel domain and misclassification rate would be increased accordingly. Two error metrics to compare the adversarial image quality with the legitimate image are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE yields the cumulative squared error between the adversarial and the legitimate image, whereas PSNR gives a measure of the peak error. The higher the value of PSNR, the higher the quality.

$$MSE = \frac{||x_i - \widetilde{x}_i||_2^2}{N} \tag{1.9}$$

$$PSNR = 10\log_{10}\left(\frac{h^2}{MSE}\right) \tag{1.10}$$

where h is the maximum fluctuation in the input image data type. For example, since we normalized all image datasets to [0,1], input images' pixels fluctuate between zero and one, so h=1. Before investigating the relation between misclassification rate and quality metrics, we recall two important properties of the matrix-vector multiplications; first, the product of an orthonormal matrix by a vector does not change the norm-2 of that vector, and second, a scalar number can take out of the norm-2 of a vector. With respect to these two properties, since $||x_i - \widetilde{x}_i||_2^2 = ||\delta D d_{ij}||_2^2$ and due to the fact that the dictionary D is an orthonormal dictionary and the δ is a scalar value, $||x_i - \widetilde{x}_i||_2^2 = \delta^2||d_{ij}||_2^2$. Equation (1.9) can be further simplified to obtain a more straightforward relation between δ and MSE or PSNR in pixel

Table 1.2: Comparing misclassification rates of directed attack over six CNN models based on proposed method (LaS) and recent method (LoF).

Model	k = 20		k = 30		k = 40	
	LaS	LoF	LaS	LoF	LaS	LoF
MobileNets	19.7	19.3	22.3	21.5	23.6	22.9
ResNet50	21.9	21.8	24.2	23.9	25.6	25.3
DenseNet121	20.0	19.2	22.3	20.8	23.4	22.3
InceptionV3	16.4	15.3	17.9	16.7	18.4	17.3
Efficient -B0	16.1	15.6	18.8	17.7	20.2	19.6
Efficient-B1	13.7	13.1	15.5	14.7	16.9	15.8

domain as follows:

$$MSE = \frac{\delta^2}{N}||d_{ij}||_2^2 = \frac{\delta^2}{N}||\hat{s}_i - \hat{s}_j||_2^2 = \frac{2\delta^2}{N}(1 - \langle \hat{s}_i, \hat{s}_j \rangle) \tag{1.11}$$

where $\langle \cdot, \cdot \rangle$ is the inner product operation of two vectors. Since both $\hat{s}_i$ and $\hat{s}_j$ are normalized vectors, their inner product equals a number belongs to $[-1, 1]$. Hence MSE can be bounded $0 \leq \text{MSE} \leq \frac{4\delta^2}{N}$. However, as we choose the two most correlated sparse vectors, their inner product is usually greater than zero. Hence, the upper bound of MSE may be smaller, i.e. $0 \leq \text{MSE} \leq \frac{2\delta^2}{N}$. This inequality shows how adding perturbation in the sparse domain can be reflected in the perturbation in the pixel domain. The value of the δ directly affects the MSE. The order of sparsity, k, only affects the inner product.

We applied the directed attack over the same six CNN models and compared the effectiveness of adding noise to the LaS components against adding noise to the LoF components. In this experiment, we used multiple values for $\{k = 20, 30, 40\}$, and we fixed the value of δ in order to have $\text{MSE} \leq 0.001$. Table 1.2 shows the results and superiority of manipulating LaS components.

As theoretically was discussed, changing δ can directly affect the perturbation level. To show this property, we trained the LeNet network [37] with 60000 training samples of MNIST dataset and achieved the accuracy of 98.2%, which means 1.8% misclassification rate over 10000 test samples. Then, we used the same test dataset and selected 6 different values for the δ and k. It leads to running 36 times, all combinations of δ and k to generate the corresponding perturbed test dataset. Then we input all these 36 adversarial sets to the LeNet classifier to observe the network's response. Figure 1.5 illustrates the effect of δ and k, PSNR, and misclassification rate of LeNet network. The left and right y-axes show the PSNR value the misclassification rate of each perturbed dataset, respectively. Solid blue lines show that PSNR decreases as delta value increases, and dash lines show that the misclassification rate increases as we increase the value of δ. We also evaluated the effectiveness of our proposed attack on the SVM classifier. Due to the computational limitation, we only used 15000 training and 3000 test samples of the MNIST dataset. After trying multiple kernels, the polynomial kernel was the best kernel to achieve the highest score for the classification. The misclassification rate of the trained SVM classifier on the benign test dataset was 5%. Then we generated adversarial sets with different levels of perturbation. Figure 1.6 shows that the SVM classifier is highly vulnerable to the proposed attack.

We compared our approach with recent work by Papernot et al. [19] which is not based on frequency domain. We used the Cleverhans library [38], and to have a fair comparison, the same CNN and parameters were used. We trained the network 10 times, and after each time, the misclassification rate of the trained model on both adversarial sets was recorded. Figure 1.7 shows that for δ =15 and k = 20, our proposed adversarial examples have a higher misclassification rate than that of the previous work, while our method has a higher PSNR which means less perceptible perturbation.

1.5 Attacking Google Cloud Vision and YOLO

To evaluate the realistic threat of LaS components perturbation, we attacked a popular online machine learning service, Google Cloud Vision. The platform provides a TFLite version that can be deployed over Android operating systems. We used a high-resolution dataset which contained 20,938 samples belonging to 10 animals "spider, dog, cat, squirrel, sheep, butterfly, horse, elephant, cow, chicken" [39]. Figure 1.8 shows the details of the trained model by Google Cloud Vision. To assess the effectiveness of our proposed attack, we downloaded its TFLite version. We randomly selected 500 test samples and added perturbation based on LaS and LoF approaches. By adding limited noise to LaS components, 132 out of 500 samples were misclassified. Also, adding noise to LoF components led to 129 misclassified samples. Figure 1.9 shows the number of required queries to fool the TFlite model based on both methods. In addition, Fig. 1.10 shows three samples and corresponding adversarial examples for MSE values equal to 0.001, 0.002, and 0.005. The first column shows the legitimate samples that are classified correctly by the classifier, the second column from the left that is closed by a green box, belongs to the adversarial examples with MSE = 0.001, the other two columns with red boxes related to the adversarial examples with MSE = 0.002 and 0.005. As defined in [25], we set the threshold of MSE$\leq$ 0.001 as a successful attack.

In addition, we applied our attack over an object detection algorithm. Object detection has been widely used by autonomous vehicles and biomedical devices. One of the fastest and most accurate object detection algorithms is YOLOv5 [40]. YOLOv5 is a one-stage algorithm that implements classification and regression tasks in a single step. Object detection algorithms implement two tasks, detection and classification. If the model fails to detect the object correctly, it may cause irreversible consequences in certain sensitive applications. In this experiment, we used *International Skin Imaging Collaboration* (ISIC)-2017 skin lesion dataset that contains 2000 training samples, 150 validation samples, and 600 test samples belonging to three skin lesion classes: *melanoma, nevus, and seborrheic keratosis.* We resized the input samples into 640x640 pixels and set two parameters as Intersection over Union (IoU) to 0.50 and confidence threshold to 0.25. We trained the model and evaluated its performance over 600 test samples. Figure 1.11 shows the performance of the trained model over test dataset. Precision measures how accurate the predictions are, while recall measures how well the model finds all the positive cases.

IoU measures the overlap between the predicted box around the object with the ground truth. The model achieved mean Average Precision (mAP) equal to 0.72 over three classes. In the next step, we randomly selected some test samples that had never been used in the training process to add perturbation and observe the model response. Our results show that by adding limited noise to the LaS components, this model predicts wrong labels with high confidence scores. In Fig. 1.12, we only showed a few adversarial examples that had been misclassified. However, the model had adversarial samples that could not detect any object. In this experiment, we set MSE$\leq$ 0.001 to generate adversarial examples. We released our code, the TFlite model trained by Google Cloud Vision, the trained object detection model, and the annotation files of ISIC-2017 dataset publicly for reproducibility [41].

1.6 Conclusion

In this work, we proposed a new approach for generating adversarial examples in the sparse domain. We show that LaS components differ from LoF components, and they belong to all frequency bands (low, middle, or high). We proposed a hypothesis that LaS components affect the decision boundaries of CNN models much more than LoF components. This hypothesis was the key to building our proposed adversarial method.

We designed a systematic experiment to support this hypothesis. By running experiments over six advanced CNN models, we empirically verified that LaS components affect the decision boundaries of CNN models more than LoF components. Then we added a limited noise to the LaS components to generate our proposed adversarial example. We evaluated the response of six advanced CNN models against our adversarial examples and compared them with recent work. Our results over MNIST and CIFAR-10 datasets unanimously support this hypothesis that adversarial examples generated based on manipulating LaS components can fool the CNN models in fewer queries than the LoF approach.

We also implemented our experiments over Animal and skin lesion ISIC-2017 datasets to evaluate Google Cloud Vision API and YOLO algorithm. Results show the effectiveness of our proposed method to fool the models mentioned above. By introducing the potential threat within this type of attack, an appropriate defense mechanism can be investigated in the future. Moreover, we used a DCT dictionary to transfer images into the sparse domain. However, there are many other ways to transfer an image into a sparse domain other than the DCT domain that can be further investigated.

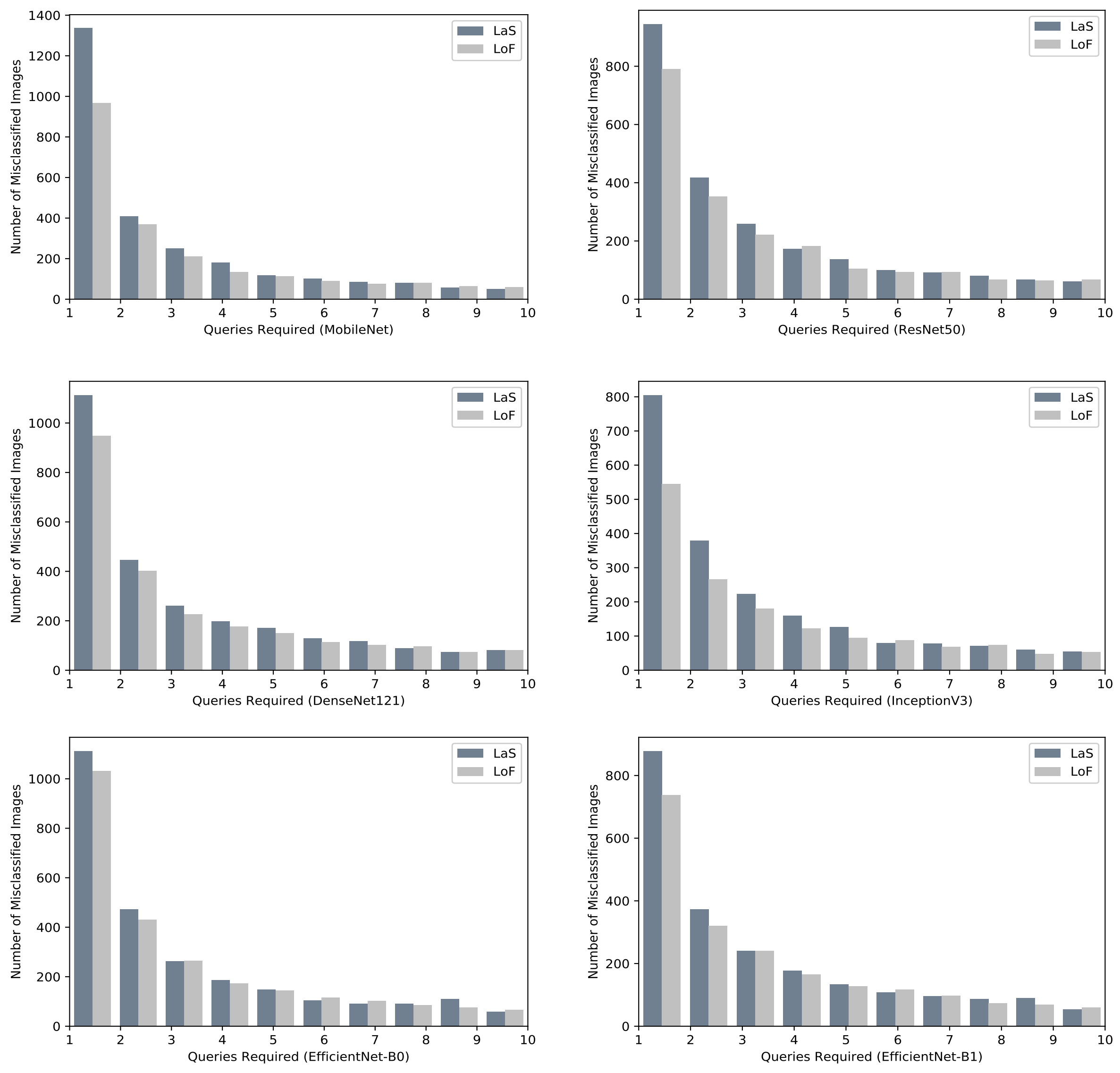


Figure 1.3: Comparing the required number of queries to fool CNN models based on proposed approach (LaS), and LoF.

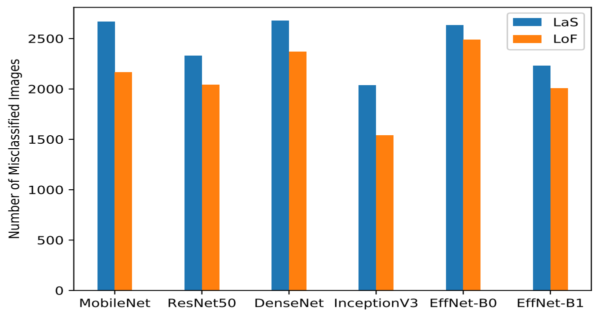


Figure 1.4: Comparing number of misclassified samples for query less or equal to 10 based on LaS and LoF.

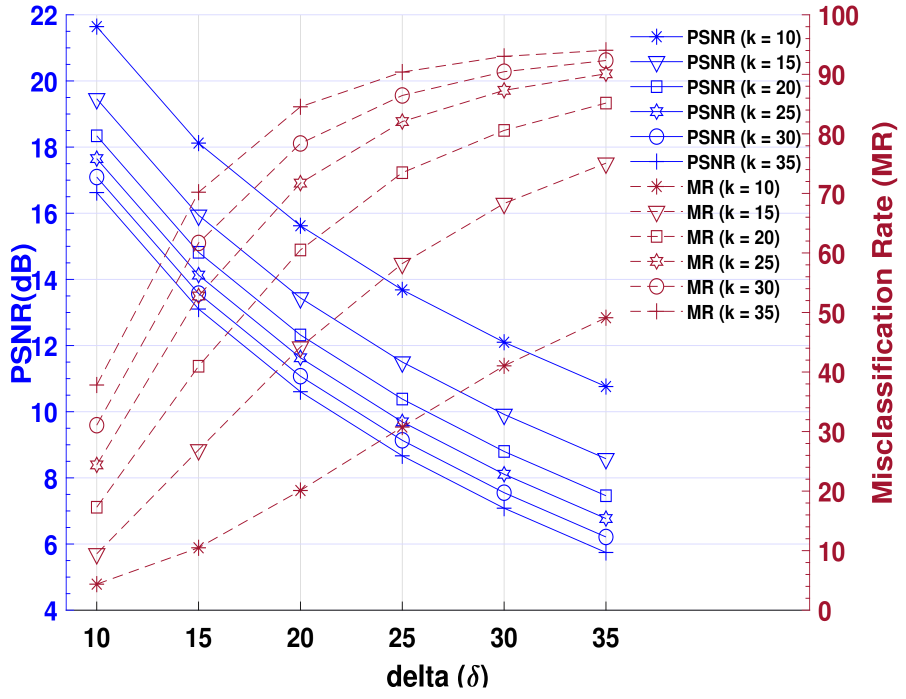


Figure 1.5: Generating adversarial examples with different level of perturbation on LeNet.

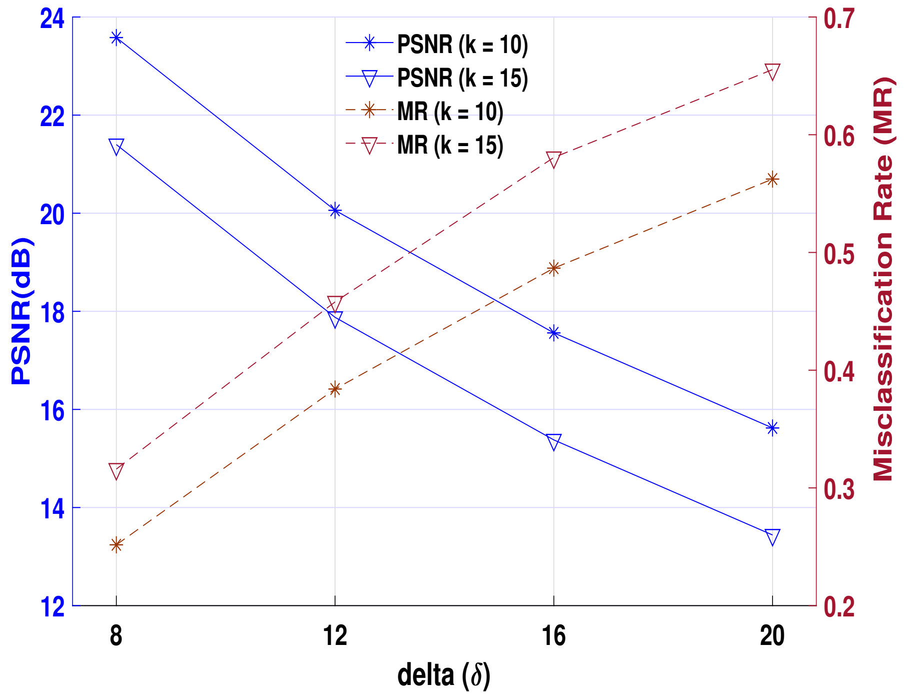


Figure 1.6: Generating adversarial examples with different level of perturbation on SVM classifier.

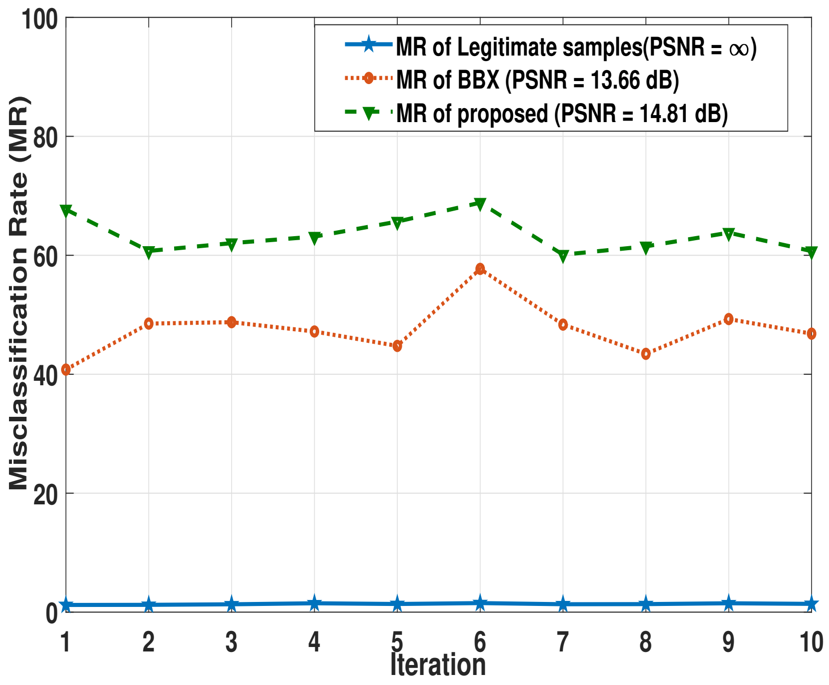
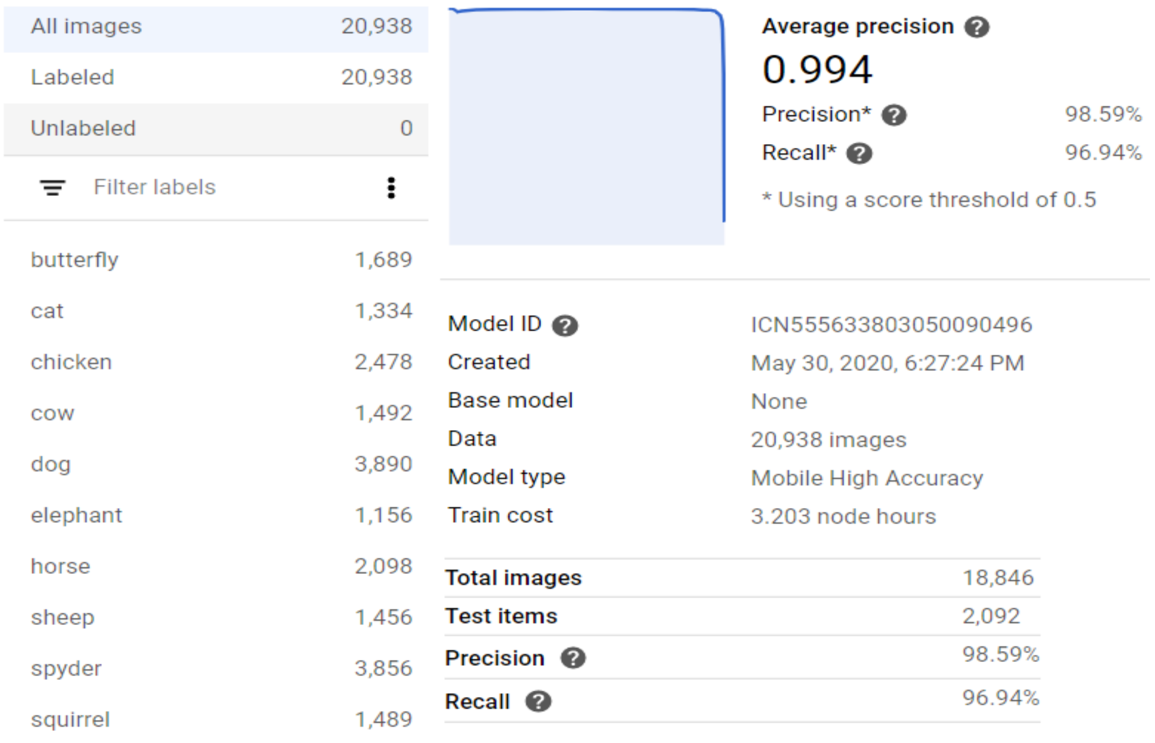


Figure 1.7: Comparing the misclassification rate of proposed method of perturbation and recent practical black-box (BBX) approach.



All images	20,938
Labeled	20,938
Unlabeled	0
Filter labels	
butterfly	1,689
cat	1,334
chicken	2,478
cow	1,492
dog	3,890
elephant	1,156
horse	2,098
sheep	1,456
spyder	3,856
squirrel	1,489

Average precision

0.994

Precision*	98.59%
Recall*	96.94%

* Using a score threshold of 0.5

Model ID	ICN555633803050090496
Created	May 30, 2020, 6:27:24 PM
Base model	None
Data	20,938 images
Model type	Mobile High Accuracy
Train cost	3.203 node hours

Total images	18,846
Test items	2,092
Precision	98.59%
Recall	96.94%

Figure 1.8: Information of dataset and trained model by Google Cloud Vision.

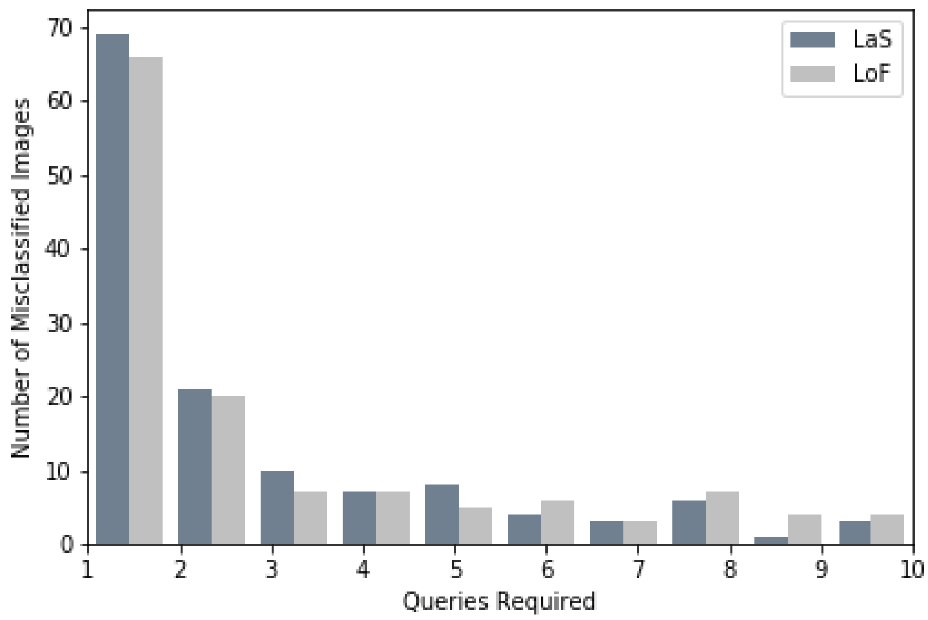


Figure 1.9: Comparing the required number of queries to fool a TFlite model trained by GoogleAPI based on proposed approach (LaS), and LoF.

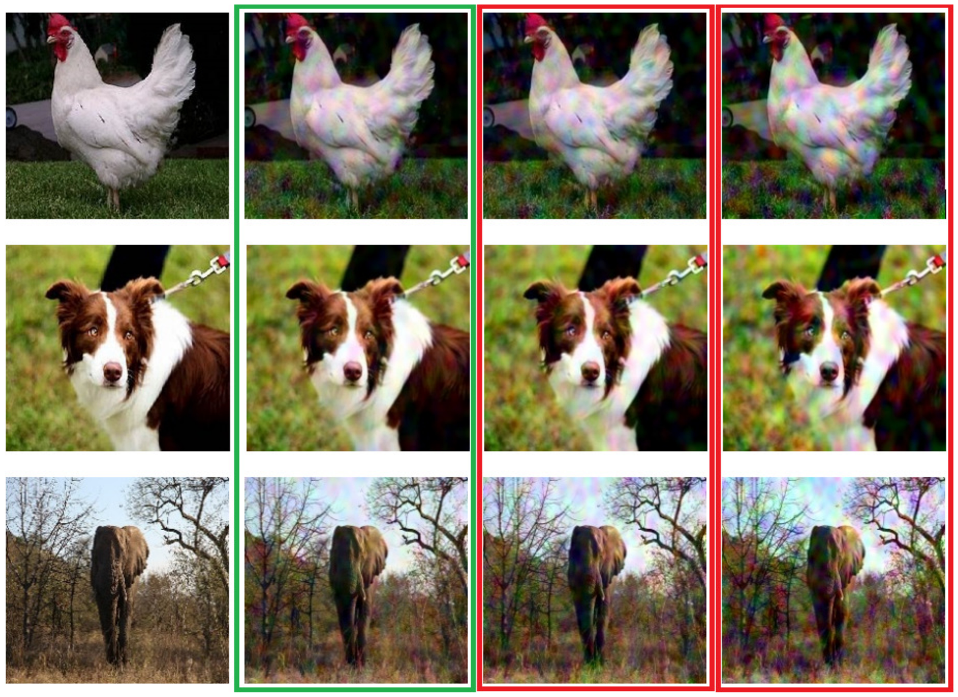


Figure 1.10: Samples of attacking Google Cloud Vision.

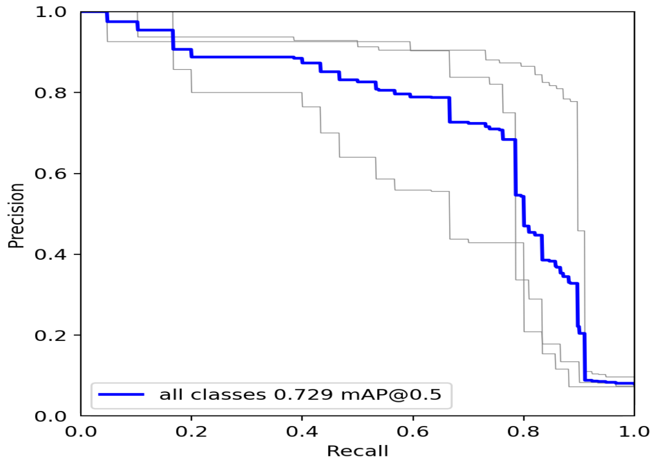


Figure 1.11: Performance of YOLOv5 over skin lesion dataset (ISIC-2017).

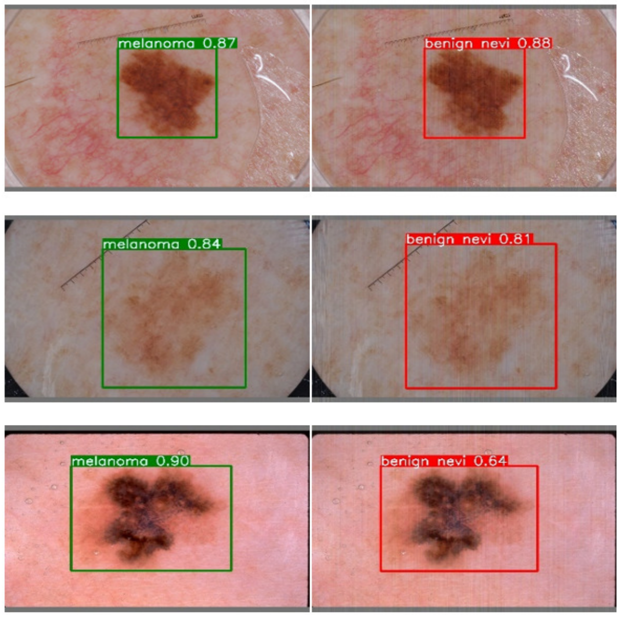


Figure 1.12: Samples of attacking the object detection algorithm (YOLOv5).

Chapter 2: Improving the Performance of Deep Learning Models

The skin lesion is one of the severe diseases, in many cases, endanger the lives of patients to a worldwide extent. Early detection of disease in dermoscopy images can significantly increase the survival rate. However, the accurate detection of disease is highly challenging due to the following reasons: e.g. visual similarity between different classes of disease (e.g., melanoma and non-melanoma lesions), low contrast between lesions and skin, background noise, and artifacts [2].

Machine learning models based on convolutional neural networks (CNN) have been widely used to automatically recognize lesion diseases with high accuracy compared to conventional machine learning methods.

In this research, we proposed a new preprocessing technique in order to extract the RoI of the skin lesion dataset. We compare the performance of the most state-of-the-art CNN classifiers with two datasets which contain (1) raw and (2) RoI extracted images. Our experiment results show that training CNN models by RoI extracted dataset can improve the prediction accuracy (e.g., InceptionResNetV2, 2.18% improvement). Moreover, it also significantly decreases the evaluation (inference) and training time of classifiers.

2.1 Introduction

With the growing advance of deep learning, numerous tasks have been solved by Artifact Intelligence (AI). Especially, the demand for AI for medical images has become emerging in recent years since, with the early detection of the disease, we can now provide better treatment plans. However, the main issue related to medical image classification is insuffi-

[2]This chapter was published in Medical Biological Engineering Computing [4]. Permission is included in Appendix A.

cient image samples. Skin is the largest organ of the body that contains lots of information about the individual's health condition, and also their identity [42]. The skin lesion is a serious disease that may lead to detrimental consequences if it does not diagnose in a proper time. There are many sources for skin image datasets, among them, the *International Skin Imaging Collaboration* (ISIC) provides public datasets that are mainly used for skin lesion classification [43, 44, 45, 46, 47]. Image segmentation is one of the most important computer vision tasks to partition an image into multiple segments. The main aim of segmentation is to locate objects of interest and its boundary to enable more efficient and effective further analysis. Segmentation has been widely investigated and implemented in many works [48, 49, 50].

There are many machine learning methods for object segmentation. One of the well-known methods for object semantic segmentation tasks is U-Net [51]. The network can be trained on both the original and augmented dataset in this method. This characteristic is primarily appropriate when the target datasets are from medical fields (mostly limited) since data augmentation enriches training samples. Also in [52], residual multitasking network achieved second place (among 28 teams) in ISBI 2016 Skin Lesion Analysis Towards Melanoma Detection Challenge segmentation task [44]. This model includes more than 50 layers with residual layers, separated into two sub-architectures for classification and segmentation. Another promising method, namely fully convolutional network (FCNs), is introduced by [53]. This deep neural architect aims to localize the coarse approximation in the early learning stage; then, the exact approximation will be learned later. Besides, the author also introduced a fusion framework to facilitate their model's performance. The final model achieved 90.66% in the PH2 dataset and 91.18% in ISIC-2016.

In [54], an end-to-end training procedure has been proposed that utilizes the Jaccard Distance loss. The model includes 19 layers trained thoroughly by their proposed loss function. Although the result is not outstanding for more challenging samples (involving hairs, badges, poor lightning condition, etc.), their approach outperforms the [44] and [53] within the same datasets. The first attempt towards multi-class segmentation on ISBI 2016 was conducted by [55], enabling segmentation with classes' information. The sequential learning method involved Faster-RCNN and U-Net in [56] also tackles the same segmentation task. In [57] a fully resolution convolutional network for learning visual representation from skin lesion images, reaching 77.11% Jaccard Index on ISIC-2017 private test set has been proposed.

In this study, we investigated the effectiveness of ROI extraction after the segmentation step to improve the performance of the classification task. We have experimented and evaluated recently developed methods of semantic segmentation so as to isolate and extract the RoI (lesion) of the images. It enables us to remove unwanted background pixels and artifacts such as hairs and badges before training CNN models.

2.2 Material and Method

We started our study with a lightweight non-training-based segmentation method, then to have a better result, we extended our study by implementing a complex training-based segmentation method. One of the non-training-based segmentation algorithms that we investigated is Otsu's thresholding [58] which clusters the background (skin) and the foreground (lesions) based on the optimal threshold from the histogram of the pixel counts. Several previous works utilize Otsu's thresholding segmentation due to its simplicity, for example, H&E staining images [59], MRI and CT scan images [60], and also melanoma lesions detection [61]. However, the main assumption of this segmentation method relies on the histogram of pixel counts, which is assumed to be bimodal distribution. Hence, the performance of Otsu's segmentation on noisy images that possess badges, hairs, and black borders is unsatisfactory.

The second approach for non-training-based segmentation is K-means clustering based on the color spaces [62]. The unsupervised cluster took three inputs: (1) two components of three color spaces: Hue which is related to the color's position on a color wheel. Cr and Cb are the blue-difference and red-difference chroma components of an image, (2) pixel-based features and (3) rough estimation of skin's boundary gained from the color-based classifier.

In both segmentation algorithms, we have used the Jaccard index for evaluating segmentation performance as follows:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

where A corresponds to the ground truth binary segmentation mask, B represents the predicted binary segmentation mask, $|A \cap B|$ is the area of overlap, and $|A \cup B|$ is the area of union [63]. We used ISIC-2017, which contains 2000 skin lesion samples with 2000 corresponding masks as ground truth, to evaluate the performance of these two algorithms [64]. Although the K-means clustering method was well-performed than Otsu's thresholding segmentation by having a Jaccard Index of 76.2% compared to 71.7%, it can still not eliminate artifacts efficiently. However, our experiments over the skin dataset showed that both non-training-based approaches could not segment images very well since artifacts like badges are more often than not segmented as skin lesions. In addition, the K-means approach does not consider the region's border. We observed that ground truth masks of the skin lesions are mostly solid closed-contour shape. However, the K-means approach does not give a solid shape. In other words, there are some small dark regions inside a detected lesion contour. This issue is because the algorithm tries to separate pixels into multiple clusters based on the mean, regardless of the position and value of near pixels. Figure 1 shows two samples of the ISIC-2017 that have been segmented based on the K-means approach. The red circles inside the lesion contour show the disability of this approach for detecting the whole lesion part.

To address this issue, and regarding the availability of 2000 masks of ISIC-2017, we followed our investigation by evaluating training-based approaches. One of the most common training-based approaches for image segmentation is "U-net" convolutional neural network [51]. U-net architecture is an evolution of the traditional convolutional neural network, which is a so-called end-to-end fully convolutional network (FCN). The architecture includes two

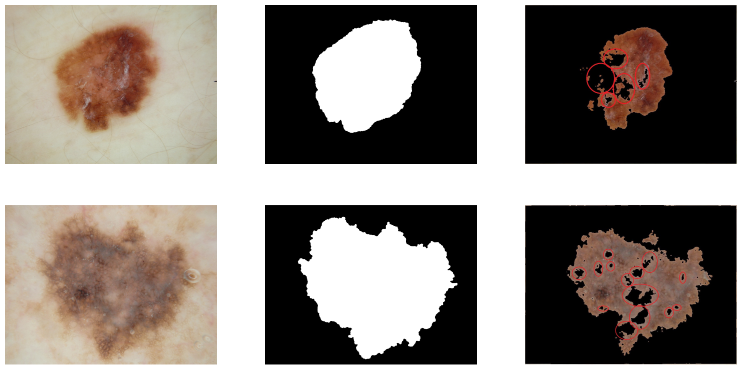


Figure 2.1: Results of K-means segmentation, from left to right: (1) raw image, (2) ground truth segmentation (3) segmented image.

parts: (1) the contraction path (the encoder) and (2) the symmetric expanding path (the decoder).

The encoder is a conventional CNN, a sequence of matrix operations (convolutional layers, max pooling, batch normalization, and so on). The main modification from conventional CNN is lied on the decoder - successive expanding path, where the upsampling operator is used instead of pooling operation. Thus, the resolution of outputs increases along with these layers. The features from the encoder are then combined with upsampled output to enable precise localization. Since fully connected layers are absent from U-net architecture, the network's outputs are segmentation maps that represent the mask of the lesion in the corresponding image.

By using the train data (images and their corresponding masks), the FCN can segment the lesion without segmenting artifacts as a lesion part. However, the database for training segmentation task is rarely available since it requires the expertise of related fields. Medical segmentation tasks often involve objects with varying size, ranging from cell nuclei [51], lung [65], retina vessels [66], and tumors [67]. Especially in dermoscopy images, the RoI often results in irregular shapes and varying sizes. Thus, the demand for a stable network that is robust to a wide scale of image sizes is necessary for further analysis. In this work, we have adopted the state-of-the-art MultiResUNet architecture, which integrated the idea of Residual Inception blocks [68]. By utilizing multiple kernels with different sizes in parallel fashion,

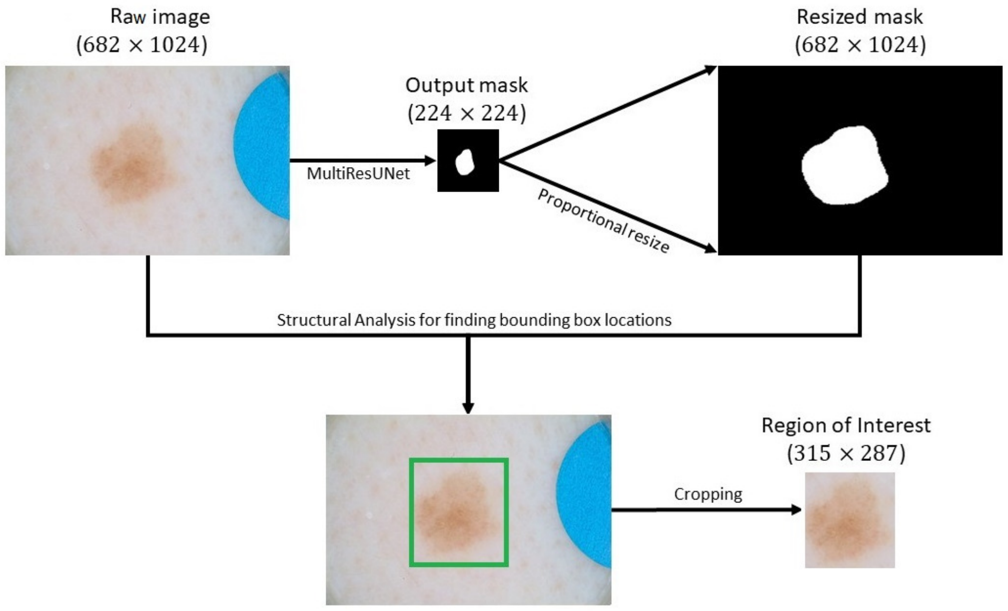


Figure 2.2: RoI extraction process: black texts show target images and its size, and red texts illustrate corresponding method used at each stage of the process.

the MultiResUNet outperforms the conventional U-Net architecture by 5.065% in skin lesion boundary segmentation. Our experimental results show that the MultiResUNet segmentation network overcomes both Otsu's thresholding and K-means segmentation method due to its strong suit built on expertise-involved training data. Within the scope of this paper, we used ISIC-2017 [64] database for skin lesion boundary segmentation. The MultiResUNet is trained by 2000 images along with corresponding masks produced by dermoscopic experts from ISIC-2017. We then selected the best segmentation model with the Jaccard Index of 80.4 for segmenting 23331 remaining images of ISIC-2019.

2.2.1 RoI Extraction

The image sizes of the skin lesion dataset are various and large. Due to the computational limitations of CNN models, the input/output layers' size of the segmentation model are fixed and smaller than that of the raw image. For example, in Fig. 2 and Fig.3 first and second

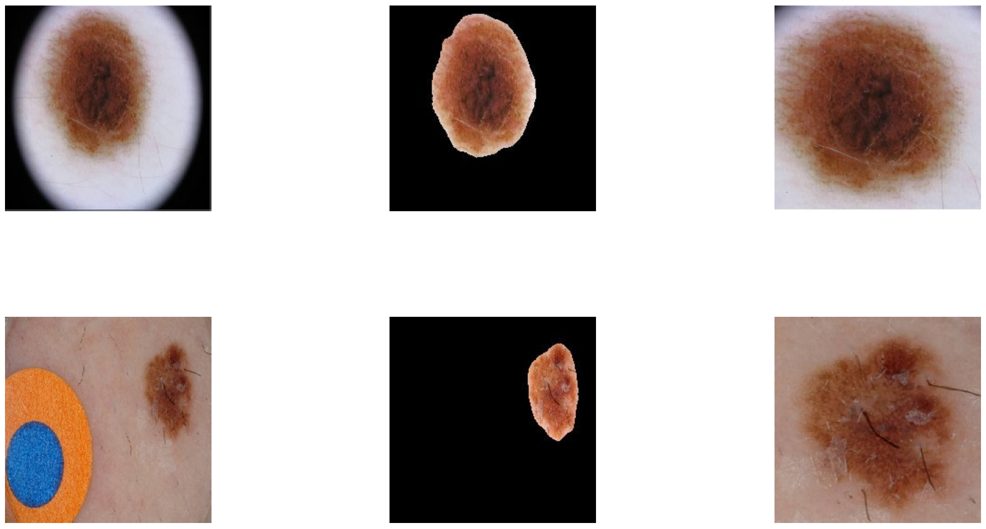


Figure 2.3: Results of MultiResUNet segmentation, from left to right: (1) raw image, (2) segmented image, and (3) RoI extracted image.

rows, the raw image size is 682×1024×3, 1024×768×3, and 1024×682×3 pixels, respectively. But the size of the input/output layers of the segmentation model is fixed to 224×224 pixels. It means, regardless of the raw image's size, the output segmented image's size is 224×224×3 pixels. Moreover, by observing the segmented images in Fig. 3, it is obvious that many pixels are black (detected as background), and only a small number of pixels is related to the lesion part. If we directly input segmented images to a classifier, those black pixels do not contribute much to the classification tasks. Also, since the legion part is highly down-sampled, some critical information could be lost. To verify our assumptions, we trained and evaluated multiple CNN models based on a segmented dataset, but their accuracy was much lower (up to 10%) than that of the raw dataset.

We developed an algorithm to extract the RoI from raw images based on derived masks to overcome this issue. As illustrated in Fig. 2, the output mask is first resized to the same size as the raw image, then contour's structural analysis was applied to detect locations of bounding rectangle box (green rectangle). After finding the rectangle of RoI, we extract this part of the raw image and input it into the classifier. In Fig. 2.3, two samples are depicted to show how our approach can focus on the most important part of the skin image. This approach can prevent to useless input pixels (black pixels) to CNN models, and the input lesion part would have more information.

In this study, the importance of extracting RoI on the performance of the CNN-based classifiers is investigated. While CNNs have been widely used for skin lesion classification [69, 70, 71, 68], to the best of our knowledge, there is no work about the effect of RoI extraction before training CNN models. In the next section, we discuss in more detail about CNNs, followed by the experimental settings and results.

2.3 Classification

2.3.1 Transfer Learning

Unlike traditional machine learning, where an expert needs to observe the target and extract good reliable features based on his knowledge, deep learning methods automatically extract reliable, high-quality features from large amounts of targets, making them more beneficial than traditional methods. Consequently, deep learning methods highly depend on mass data since they need a large amount of data to have a reliable comprehension of the patterns of the data set. The more data set a deep learning network has, the bigger it should be to extract well-behaved features. It means that to achieve an outstanding performance from a deep neural network, it needs a large amount of data that requires an extensive network to understand its patterns.

In deep neural networks, some of the final layers are responsible for making a decision related to the task, and the rest of them can be used to extract high-level features. Lack of sufficient data is one of the main problems that researchers usually face when training a model on specific data. This problem can be more severe in biomedical image classification tasks since it is much harder to find a large amount of a high-quality data set. Transfer learning addresses this problem and is a solution. In traditional machine learning, we should consider that training data must be independent and identically distributed with the test data. However, transfer learning assumes that training and test data do not need to be independent and identically distributed entries. It means that for a specific task, the network is not required to train from scratch, which has two specific benefits, first, it eliminates the requirement of accessing a big dataset, and second, it reduces the time of training the network.

CNN pre-trained models are usually trained on large image classification tasks. Convolutional layers are responsible for extracting high-level features from an image, while dense layers must decide on those features. There are two kinds of transfer learning: feature extraction and fine-tuning. In the former, the convolutional layer parameters are frozen during back-propagation and are used to extract features on a new data set, and the new dense layer is added to fit the network for the new data set. After adding a new dense layer, relax back-propagation will be done on the whole parameters for tuning them with the new dataset. Pre-trained models by ImageNet have been widely used for skin lesion classification[72, 73, 74]. We also tested this property and found that if we initialize the models with ImageNet pre-trained wights, the training process converges in fewer epochs while maintaining higher accuracy. Also, we used data augmentation by randomly rotating the training images up to 90 degrees, and flipping them horizontally.

2.3.2 Deep Learning Models

Since 2012, when Alex Krizhevsky *et al.* introduced the AlexNet[75], CNN models which were not able to absorb attention came back to the play and in the next few years, many researchers and experts tried to come up with new deep neural networks in the similar way to improve the accuracy on different tasks[76, 32, 77, 34, 78, 36, 33, 79]. We have used some of these pre-trained networks via transfer learning in our work to evaluate the effectiveness of our proposed method. We have used InceptionResNetV2[76], Xception[77], InceptionV3[34], DenseNet[36], ResNet-152[33], and VGG19[79] which each of them has different architecture in the number of fully connected layers and convolutional layers.

These artificial models have tried to look at different problems and are designed by some experts since it requires a suitable selection of architectures that need high-quality knowledge in machine learning, as well as it is a tedious and time-consuming task. Moreover, different architectures should be designed for different targets to get a better result. However, some works named the neural architecture search(NAS) have been introduced recently to address this problem and try to find a good architecture for a certain target automatically, which is logically suitable for different types of image classification. In our work, we also tried to use some of these networks, which are trained well on the ImageNet data set to evaluate our work. We have used two networks EfficientNet[32] and NasNet[78] which both are automatically designed by Google brain team members.

2.4 Experiments and Results

We used ISIC-2019 dataset that contains 25331 dermoscopic images belong to 8 classes: *melanoma*, *melanocytic nevus*, *basal cell carcinoma*, *actinic keratosis*, *benign keratosis*, *dermatofibroma*, *vascular lesion*, and *squa-mous cell carcinoma*. On the other hand, the ISIC-2017 dataset contains 2000 samples with masks for skin lesion segmentation and classification. These 2000 samples of ISIC-2017 are exactly available in ISIC-2019 as well. We used those 2000 samples and their masks to train the MultiResUNet model, and we did not involve them in the evaluation. We set the input/output size of the segmentation model to 224×224 pixels. After training MultiResUNet model for 50 epochs, we generated 23331 segmented images(2000 samples out of 25331 were excluded). The average required time for segmenting each image was 11 milliseconds (Ms). Then, we applied our RoI extraction algorithm in order to focus on the main information of the image. After doing so, two sets of images were generated for our experiments: one contained 23331 raw images and the other had 23331 corresponding RoI extracted images. We randomly split the 23331 samples into three sub-sets, training (18890 samples), validation (2101 samples), and test (2340 samples). We applied the same data augmentation from the Keras framework over both training datasets by randomly rotating the images up to 90 degrees and horizontally flipping them.

Python was used as the programming language. We used Keras, a high-level neural networks API written in Python and can run on top of TensorFlow. All models were directly selected from Keras documentation, and the default input size of the first layer was chosen according to the Keras documantation[80]. Keras framework provides models that can be converted to TensorFlow lite (TFLite) format. TFLite models can be deployed over the android operating system. We used a single GPU (Nvidia GeForce GTX 1080 Ti with 11 GB GDDR5X memory) for all of our experiments (training and evaluation).

We have done several experiments in order to demonstrate the effectiveness of the segmentation and RoI extraction before the classification. To have a fair comparison, all input and hyperparameters were set the same for each model while training raw or segmented datasets. We used a learning rate of 0.001, batch size 8 for the NasNet model, and batch size 16 for the rest of the models. Default input size of each model was used (refer to Table 2.1). Each model was trained for 50 epochs and 1000 steps per epoch.

In Table 2.1, we evaluated the accuracy of each trained model over the test dataset. The results show that extracting the RoI prior to classification can improve the accuracy of skin lesion diagnosis. Table 2.1 only presents the results based on the default size of each network where we obtained the best accuracy of each model. However, we trained each model with a smaller size as well. The smaller input size gave lower accuracy but higher improvement.

Since skin lesion contains eight unbalanced classes, overall accuracy may not completely convey the effectiveness of our approach. Hence, to evaluate the impact of our approach over each class separately, we used an F1-score which can yield a more realistic measure of the classifier's performance. It avoids being misled by the average accuracy that can be wrongly obtained from a very poor precision or very high recall. Figure 2.4 shows the F1-score of two diseases, *'melanoma'* and *'nevus'*, and also weighted-average over all classes. In this figure, classifiers' indices are as follows: #1: InceptionResNetV2, #2: Efficient-B7, #3: Xception,

Table 2.1: Effect of segmentation on the accuracy of CNN classifiers (%).

Model	Input size	Raw data	RoI extracted data
InceptionResNetV2[76]	299 × 299	86.75	88.93
Efficient-B7[32]	224 × 224	84.87	87.95
Xception[77]	299 × 299	86.03	87.74
InceptionV3[34]	299 × 299	87.09	87.39
NasNet[78]	331 × 331	85.85	86.24
DenseNet[36]	224 × 224	85.68	86.03
ResNet-152[33]	224 × 224	84.15	85.17
Efficient-B0[32]	224 × 224	81.75	84.87
VGG19[79]	224 × 224	80.17	82.61

#4: InceptionV3, #5: NasNet, #6: DenseNet, #7: ResNet-152, #8: Efficient-B0, #9: VGG19. The proposed approach can improve the performance of almost all classifiers.

In Table 2.2, the required time for predicting the label of each input sample has been reported. It is evident that the inference time of segmented data is much lower than that of the raw data. To have stable results, we repeated our experiments ten times over the test dataset, and then the average of results was calculated. In Table 2.3, the required time for training each model has been evaluated for both datasets. It shows that segmented data can be trained in a shorter time. Although training and generating segmented images take time, in certain cases, we generate a dataset one time and use it for training many models. For example, in ensemble learning, many models are used to calculate the best accuracy [81]. Using a segmented dataset would significantly decrease the time of training and, as a result, save more power and computational resources. By using segmented images, the required times for training all models would be decreased, but the required time for the evaluation would be decreased accordingly. The size of the raw dataset (23331 samples) is 9.4 GB (gigabyte), while the size of the segmented dataset is 1.4 GB. On overage, the size of a segmented image is less than one-sixth of a raw sample. This property would be important if we need to send the data over a communication channel. For example, in a remote classification task, a client may send the image to a remote server for doing more

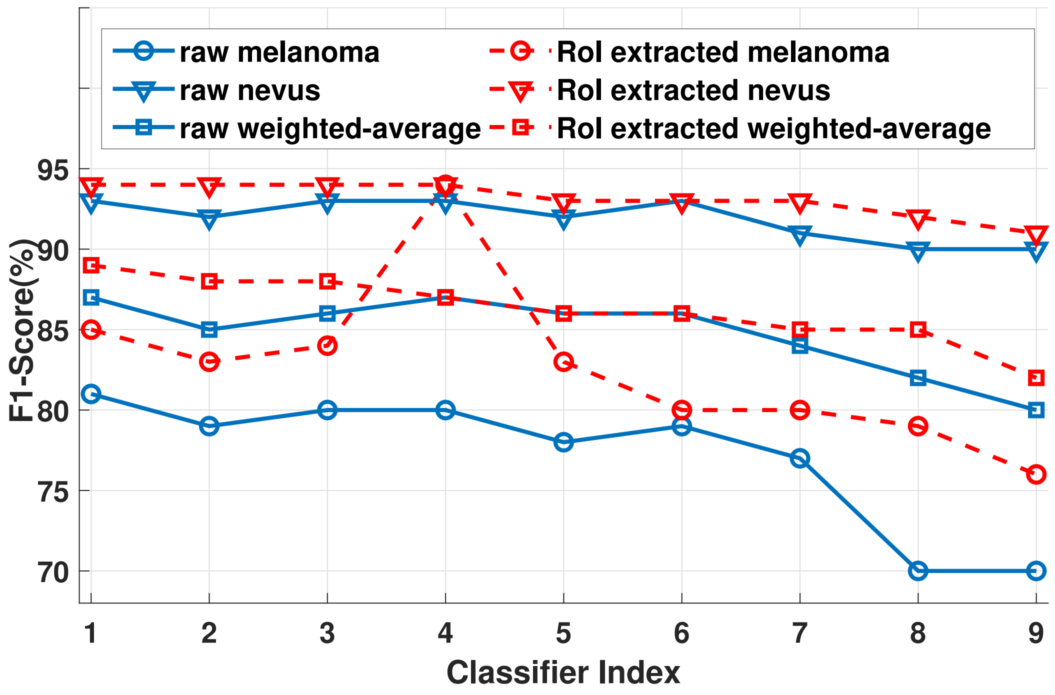


Figure 2.4: F1-score of two classes: *'melanoma'*, *'nevus'*, and weighted-average of all classes.

reliable classification. By doing a segmentation over the client-side (e.g., android device), we would be able to reduce the communication bandwidth.

2.5 Case Study: Effect of Compression on the Performance of the Skin Lesion Classification

Image compression is an important type of data compression applied to digital images to decrease storage or transmission costs. Image compression may be lossless or lossy. Lossless compression methods are preferred for archival purposes and often for medical imaging. In many IoT resource-constrained devices, images are sent to a remote server through a communication channel in order to apply appropriate post-processing. A straightforward scenario is sending raw images to the server. However, it may take more bandwidth uploading time. By applying an appropriate compression technique, we can reduce the size of the image, thereby decreasing the upload time and required communication bandwidth. There

Table 2.2: Effect of segmentation on the inference time of CNN classifiers (Ms).

Model	Input size	Raw data	RoI extracted data
InceptionResNetV2[76]	299 × 299	22	12
Efficient-B7[32]	224 × 224	20	14
Xception[77]	299 × 299	24	10
InceptionV3[34]	299 × 299	22	8
NasNet[78]	331 × 331	30	23
DenseNet[36]	224 × 224	19	8
ResNet-152[33]	224 × 224	21	11
Efficient-B0[32, 35]	224 × 224	19	7
VGG19[79]	224 × 224	21	9

Table 2.3: Effect of segmentation on the training time of CNN classifiers (Hours).

Model	Input size	Raw data	RoI extracted data
InceptionResNetV2[76]	299 × 299	16.2	9.6
Efficient-B7[32]	224 × 224	17.4	11.7
Xception[77]	299 × 299	11.8	6.7
InceptionV3[34]	299 × 299	12.8	10.5
NasNet[78]	331 × 331	13.3	12.7
DenseNet[36]	224 × 224	11.4	9.3
ResNet-152[33]	224 × 224	10.5	7.1
Efficient-B0[32]	224 × 224	6.5	3.5
VGG19[79]	224 × 224	12.7	5.3

are many image compression techniques; however, only a few are energy efficient and require very limited computational resources. In this study, we suppose a resource-constrained IoT device has generated the image, such as a smartphone; as a result, we propose using a method to be fast, energy efficient, and deployable over IoT devices. Compression should not affect images' quality and image classification performance. Compressed sensing was one of the best choices that could meet our requirements.

Compressive sensing (CS) is a sampling technique for efficiently sampling a signal by solving under-determined linear systems. It takes advantage of the signal's sparsity, and fewer measurements than the Nyquist rate can effectively represent the signal. For instance, given a vectorized image signal x $\in R^N$ and an orthogonal basis $\Psi \in \mathbb{R}^{N \times N}$, then one can map the image signal to sparse domain via, $= \Psi s$, where $\in \mathbb{R}^N$ is a sparse vector with k (k << N) nonzero entries. In other words, is a sparse representation of under the chosen pre-defined dictionary. The compression phase in CS provides the measurement vector through a linear operation as given below:

$$y = \Phi x = \Phi \Psi s \tag{2.2}$$

where, $y \in \mathbb{R}^M$ is the measurement vector and $\Phi \in \mathbb{R}^{M \times N}$ is the measurement matrix. For simplicity, let $A = \Phi\Psi$. $A \in \mathbb{R}^{M \times N}$ is a rectangular matrix, sometimes referred to as "total" dictionary in the CS literature. For exact and stable recovery of sparse signal, *restricted isometry property* (RIP) is a sufficient condition. RIP is satisfied if there exists a restricted isometry constant (RIC) δ_K , $0 < \delta_K < 1$ such that

$$(1 - \delta_K)\|s\|_2^2 \leq \|As\|_2^2 \leq (1 + \delta_K)\|s\|_2^2 \tag{2.3}$$

where δ_K denotes the isometry constant of a matrix A, and its value belongs to a set of real numbers between zero and one. But, checking the RIP condition of a matrix or calculating the value of its isometry constant is difficult to verify. Hence, conditions that lead to RIP were proposed. Another condition, which is easier to verify in practice, is the requirement that measurement matrix Φ must be incoherent with the sparsity basis Ψ. Mutual coherence μ between Φ and Ψ is defined as follows:

$$\mu(\Phi, \Psi) = \sqrt{N} \max_{i,j} \frac{|\langle \phi_i, \psi_j \rangle|}{\|\phi_i\|_2 \|\psi_j\|_2} \tag{2.4}$$

where $\phi_{i\in\{1,\ldots,M\}}$ and $\psi_{j\in\{1,\ldots,N\}}$ respectively represent the row vectors of Φ and the column vectors of Ψ. The coherence measures the maximum correlation between the two matrices. Smaller coherence can lead to better signal reconstruction performance and higher quality. Since $\mu \in [\,1, \sqrt{N}\,]$, the matrices Φ and Ψ are incoherent if $\mu(\Phi, \Psi)$ is closer to one, which corresponds to the lower bound of μ.

A step called the *recovery process* reconstructs the input signal from the measurement vector by solving the equation (2.2). Since is a rectangular matrix (M < N), the problem formulated in equation (2.2) is ill-posed and has infinite solutions. However, based on the knowledge that has a sparse representation regarding a basis Ψ, the recovery process can be performed in two steps. The first step finds the sparse vector $\tilde{s}$ by solving the following optimization equation. Once the vector˜has been obtained, the second step reconstructs the original signal as follows:

$$\tilde{x} = \Psi \tilde{s}. \tag{2.5}$$

Various methods have been proposed to find an appropriate solution to the equation leading to numerous recovery algorithms such as Basic Pursuit, StOMP, OMP, CoSAMP, Belief Propagation, and SL0. Compressive sensing (CS) takes advantage of the potential sparsity within signals, and through a non-adaptive linear measurement process, it can generate compressed samples. In this approach, the compression phase is highly fast, energy-efficient. Figure 2.5 shows one sample of skin lesion that has been compressed via compressed sensing method for compression ratio (CR) of 4:1.

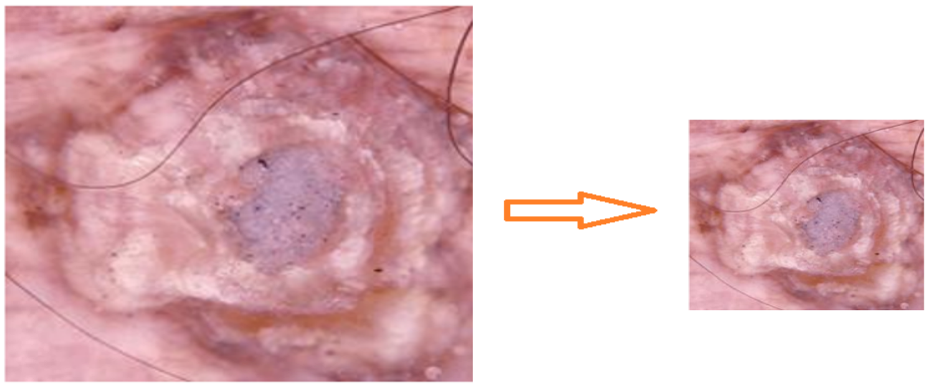


Figure 2.5: Compressing one sample with CR = 4:1.

In the next step, we assessed the effect of compression on the performance of skin lesion classifiers. With this regard, we compressed and sent the whole ISIC dataset to a remote server. Then, we recovered all images and trained different CNN models to evaluate the performance of classification tasks. Figure 2.6 shows the details of our experiments. We compared our results with the case that there is no compression. As illustrated in Table 2.4, if we compress images in edge devices and send them to the remote server, then making a recovery before evaluating the ML model, it would improve the accuracy. In other words, the compression-decompression process can reduce the noise of images, as CNN models would be trained with less noisy data.

Furthermore, the test data would have less noise and as a result, the diagnosis would be more precise. Also, we compared this CS-based compression with SVD-based compression technique. Our experiments show that our proposed compression method can outperform the SVD-based approach. As shown in Table 2.5, for a given compression ratio, CS-based compression can maintain the accuracy much better than that of the SVD-based compression. Figure 2.7 shows the effectiveness of using CS in terms of required computational resources in both the client-side and server-side. We compressed and recovered 2340 skin lesion samples based on the CS method. The average time for the compression phase is 0.004 seconds, while the average time for recovery is 2.33 seconds. Regarding our objective of having resource-constrained devices in clients, this experiment shows that CS can opt as a compression

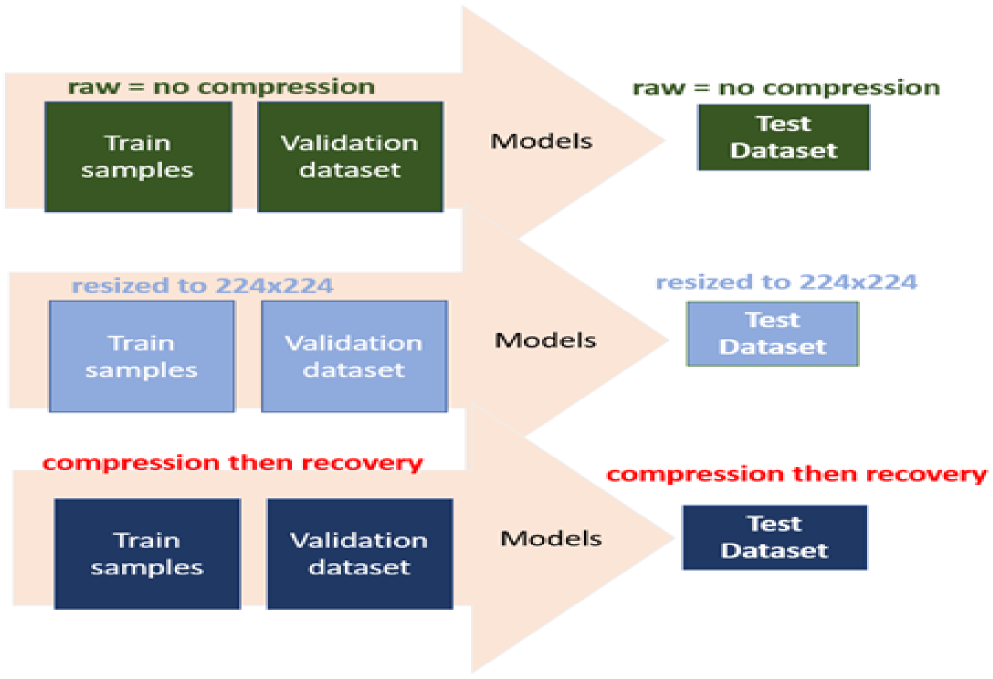


Figure 2.6: Training and testing each model with their corresponding datasets

Table 2.4: Effect of compression on the accuracy of CNN models.

	Raw	CR=2	CR = 4	CR = 6	CR=8	CR = 16
Resnet-152	86.88	87.61	88.59	87.18	87.09	86.67
EfficientNet-B0	88.83	88.97	89.10	88.08	87.91	86.71
EfficientNet-B1	89.31	89.0	89.61	89.16	88.50	87.05
EfficientNet-B2	88.80	89.49	89.23	88.33	88.46	87.95
EfficientNet-B3	90.26	89.79	90.30	89.57	88.25	87.61

method. Furthermore, we compared our method with that of the SVD based compression. The average compression time for SVD-based was 0.01 seconds which is two and half times larger than that of CS-based compression.

2.6 Conclusion

This study proposed a new preprocessing technique to separate the RoI section from the unwanted background of skin lesions. To this end, we used one of the state-of-the-art segmentation algorithms, MultiResUnet, to segment the skin lesion image. Then, we found the bounding box around the lesion and cropped that part of the image. We applied this

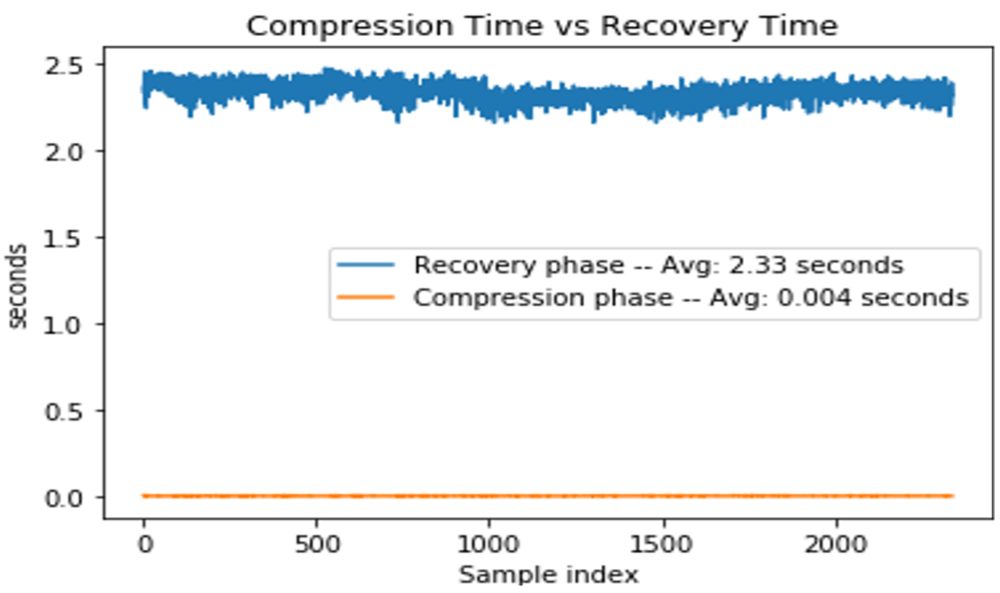


Figure 2.7: Elapsed time of compression phase vs recovery phase of cs-based method

Table 2.5: Comparing different compression methods.

	CR = 4		CR=16	
	CS	SVD	CS	SVD
EfficientNet B0	89.10	88.59	86.71	83.72
EfficientNet B1	89.61	88.33	87.05	85.00

preprocessing over the whole ISIC-2019 except for 2000 samples of this dataset used for training the MultiresUnet model. This preprocessing enabled us to only input the essential part of the image to the classifiers for both training and evaluation steps. Our investigation over different CNN models showed that if we train models with RoI extracted dataset, the accuracy of models would be increased, and the training and inference time would be dropped. We remove unwanted background and only input the most important part of the skin image to the classifiers. Our results were based on Keras models in this study, and we used ImageNet pre-trained weights to train all classifiers. We trained the models based on their default input resolution size (based on Keras Documentation). As the discussion above revealed, focusing on the RoI part of an image can improve the performance of skin lesion classification. Instead of the segmentation technique, an object detection algorithm can also be used to find the RoI of skin lesions. However, there is no available annotated dataset for training object detection algorithms. But, converting the mask labels of ISIC-2017 to the annotated dataset, would enable us to train an object detection algorithm. It can speed up the RoI extraction phase and potentially improve the final accuracy of classifiers.

Chapter 3: Lossless Privacy-Preserving Image Denoising

Image denoising aims to obtain the original image from its noisy measurements. While the quality of image denoising has been increasing over the years, the complexity and the required memory in order to implement the denoising task have also been increased accordingly. Along with such advancements, and due to the unlimited computing resources available in the cloud, trends to transfer the image denoising task to the cloud have grown over the past years. However, it is still quite challenging to utilize cloud-based resources without compromising users' data privacy while maintaining the quality of image denoising. In this chapter, we propose a novel lossless privacy-preserving image denoising approach that protects the users' privacy and preserves the quality of the denoising task concurrently. Our proposed approach is suitable for computationally constrained devices such as many IoT devices. In this method, we use two random keys to permute and perturb the noisy image patches. The cloud service provider implements the denoising task on the encrypted signal. After denoising, the output signal is still encrypted, and the real user who has access to the keys would be able to decrypt the denoised image. We evaluate the security of this method against known-plaintext, brute-force, and side-channel attacks. In addition, we theoretically prove the lossless property of this method. To verify the applicability of this approach, we implemented our experiments on multiple real images, and two well-known evaluation metrics were used to compare our results with the baseline.

3.1 Introduction

With the recent advancements of the internet of things (IoT) and computing technologies, large-scale image datasets obtained from various applications such as medical imaging equipment, remote surveillance and traffic devices, smartphones, and satellites have been growing. For example, in the context related to multimedia healthcare [82], images are intermittently generated from IoT-enabled and wearable medical imaging devices [83]. When such data explosion is in high demand to store, process, and manage images via cloud resources. [84, 85]. One of the most prominent and long-standing image processing techniques is image denoising. Noise is a ubiquitous signal that can affect any image sensing process and degrade the image's visual quality. In specific biomedical applications such as autonomous skin lesion segmentation [86], nail fungus disease detection [87], or magnetic resonance imaging [88], it may cause wrong diagnosis. Also, a noisy image can mislead the state-of-the-art deep networks deployed over autonomous vehicles by making wrong decisions [89].

The main objective of image denoising is to recover an original image from its noisy version. In the literature, there exist numerous image denoising techniques. Among them, Patch-based image denoising methods provide the highest quality [90, 91, 92]. In this chapter, we consider the classic patch-based image denoising problem, which is capturing an ideal original image I in the presence of the additive white Gaussian noise V with standard deviation σ [91, 92]. As shown in Fig. 3.1, the noisy measured image Y can be obtained as follows.

$$Y = I + V \tag{3.1}$$

Restoring accurate I from Y is a challenging problem, i.e., zeroing as much noise as possible while preserving the visual details in Y. A common approach to implement an effective denoising algorithm is generating noisy image patches. For this purpose, an overlapped sliding window (solid red color window) is used to generate all possible noisy patches (Fig. 3.1). One of the well-known approaches for patched-based image denoising is K-singular value decomposition (K-SVD) [92]. This algorithm takes advantage of image patches' sparsity to restore the noisy image.

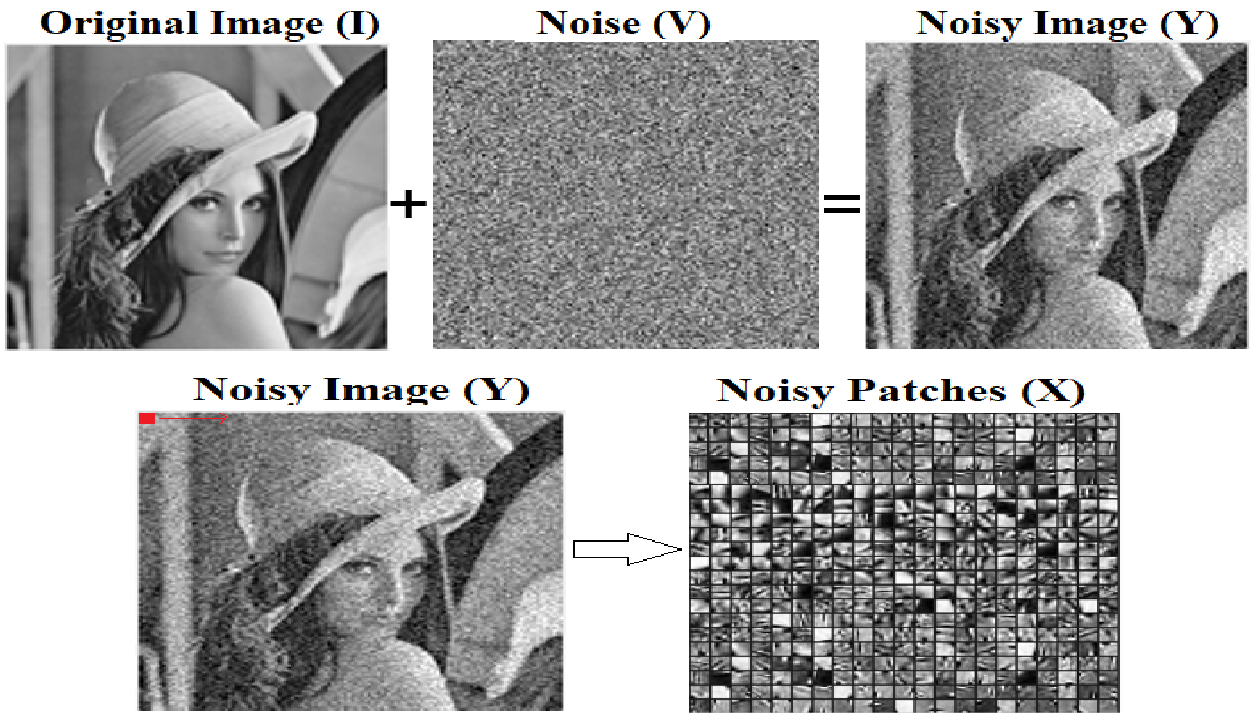


Figure 3.1: Classic patch-based image denoising problem.

Over the years, other denoising methods based on supervised deep-learning [93, 94, 95, 96, 97] [93, 94], EPLL [98], BM3 D [99], WNNM [100] could surpass the K-SVD method. However, very recently, deep K-SVD denoising was proposed by Scetbon et al.[90] and could outperform the state-of-the-art denoising algorithms. They trained a deep model with the same K-SVD computational path in order to obtain optimized denoising. In other words, they try to use deep architecture to train a few parameters so as to improve the K-SVD approach.

While the quality of denoising has been increasing over the years, the complexity and the required memory to execute denoising processes have been increased accordingly[90]. For example, given a noisy image, thousands of patches of that image should be generated to input the denoising algorithm [90]. Also, many of the current methods use a deep architecture which requires a strong enough processor to train or test processes.

The usages of edge devices with limited power and computational resources such as smartphones or surveillance cameras have been rapidly increased. On the other hand, cloud or powerful servers have been steadily backing up the edge devices by providing robust storage and computational resources.

Outsourcing is a way to shift away from the service from the user (edge device) with limited resources to the cloud environment with strong computational resources in order to store data or solve complex problems[101, 102]. However, sending users' data to the cloud may expose the users' privacy. For example, biomedical images contain private information of users that should not be revealed [103, 104].

Privacy-preserving outsourcing means implementing a process on the cloud resources while protecting the users' private information. The problem of privacy-preserving outsourcing is similar to the concept of zero trust framework, i.e., we do not trust any party, particularly the cloud environment, for accessing the real data (noisy image patches) [105]. This is an ideal case from the security perspective. However, the cloud requires data to implement our requested task.

There are many strong data encryption methods, but such methods generate encrypted data that cannot be used (or harder to be used with) to implement the denoising task. Therefore, before proposing or using any data encryption method, we must have knowledge about the task's properties and requirements, then choose an appropriate encryption method that preserves privacy and enables the cloud to perform the task accurately. This process is similar to homomorphic encryption methods, where the cloud is able to execute the task over encrypted confidential data without decrypting them [106].

On the other hand, some non-training-based denoising methods yield moderate denoising results and can be implemented on edge devices. However, when a client demands high-quality denoising and pays the cost of outsourcing in terms of time and communication bandwidth, he does not expect to lose quality due to the applied privacy-preserving method. Otherwise, the client may not choose that privacy-preserving method, or even the problem of shadow IT may come up.

In this study, we propose a method that not only enables the cloud to perfectly implement the image denoising task but also protects the private data of users. The summary of our contributions are as follows:

- We propose lossless privacy-preserving image denoising, which maintains the denoising quality and privacy of users concurrently.
- We theoretically prove the lossless property of this method. We also verify this property by implementing our method on multiple noisy images.
- We show the robustness of the proposed approach against known-plaintext, brute-force, and side-channel attacks.

3.2 Related Work

In the literature, there are some works particularly related to privacy-preserving image denoising in the cloud environment [107, 108, 109, 110]. Before discussing those works, we define two terminologies to categorize different privacy-preserving outsourcing methods:

- *Lossless privacy-preserving outsourcing*: refers to the case when the quality of image denoising with and without applying the security model is the same. In other words, image denoising quality based on privacy-preserving outsourcing is the same as regular outsourcing.
- *Lossy privacy-preserving outsourcing*: is the case when cloud does the image denoising task with some degradation compared to the case of regular outsourcing due to the applied security model.

In [107], a privacy-preserving image denoising service on the cloud-side has been introduced. They used an image denoising technique based on deep neural networks (DNNs), and via lightweight secret sharing and garbled circuits were able to preserve the privacy of image content on the cloud. Their security design could not maintain the perfect quality of denoising service; hence it falls into the lossy privacy-preserving outsourcing category.

Zheng et al. [108] proposed DNN-based image denoising with security features based on the homomorphic encryption model. This method requires training a DNN model, and the user should stay online while interacting with the remote server for the denoising process. Furthermore, their method is a lossy one as it cannot achieve the perfect denoising process. Also, Zheng et al. in [109] used encrypted external databases in the cloud environment and proposed a privacy-preserving image denoising method. They used a real-world image dataset and achieved a comparable denoising quality, yet not perfect denoising quality.

Hu et al. [110] proposed a double-cipher architecture to implement privacy-preserving image denoising based on nonlocal means in encrypted images. They used a privacy-preserving transform to generate one ciphertext, and a Paillier scheme to generate another ciphertext. Paillier is a partially homomorphic encryption system that enables two types of operation: multiplication of ciphertext by a plaintext number, and addition of two ciphertexts. They used the first cipher for the nonlocal search and the second cipher to enable the mean filter. Their proposed security scheme does not affect the time of denoising compared to regular denoising, yet cannot achieve the perfect quality of denoising service.

While all aforementioned works tried to securely implement the denoising task, they failed to achieve the exact quality of denoising compared to the regular outsourcing. In this study, we use a well-known patch-based image denoising technique and propose a lossless privacy-preserving image denoising method. We theoretically prove the lossless property of this method, and our experiment results verify that this method achieves the exact quality of denoising compared to the regular denoising task. We show how effectively this method can preserve the privacy of the users' data without losing the quality of image denoising tasks. To the best of our knowledge, this is the first lossless privacy-preserving image denoising method. The rest of the chapter is organized in the following manner. In the next section, the preliminary information about image denoising is introduced. In section 3.4 our proposed lossless privacy-preserving method is discussed, followed by the proof of lossless property and security analysis. Section 3.5 contains the experimental results to verify the applicability of this approach. Finally, section 3.6 concludes the chapter.

3.3 Preliminary

3.3.1 Image Denoising

One of the well-known image denoising methods is based on sparse and redundant representations of image patches over trained overcomplete dictionaries[91, 90]. By transferring noisy signals into a sparse domain, we can separate some parts of noise from the noisy signal. Basically, transferring signals into a sparse domain enables us to separate noise from the signal of interest. In Fig. 3.2 and Fig. 3.3, a simple example to elaborate this property of sparse representation is shown. By adding white Gaussian noise to the original signal $x(t) = 0.7\ \sin(100\pi t) + \sin(240\pi t)$, and by using the Discrete Cosine Transform (DCT), we transformed the signal into the sparse (frequency) domain in which the effect of noise can be clearly identified. The denoised signal can be obtained by putting a threshold in the sparse domain and then transferring the signal back into the time domain.

In the aforementioned example, a fixed dictionary (DCT) was used. However, for more complicated and practical signals such as a noisy image, training a specific dictionary in order to have a more sparse representation can lead to more noise removal. With this regard, patch-based image denoising algorithms have been proposed as an effective image denoising method. The main objective function of patched-based image denoising is the following optimization problem [91]

$$\min_{\Psi,S} \|X - \Psi S\|_F^2 \quad \text{s.t} \quad \forall_i \|s_i\|_0 \leq T_0 \tag{3.2}$$

where $X \in \mathbb{R}^{np}$ contains p noisy image patches of size $\sqrt{n} \times \sqrt{n}$ pixels that have been reshaped to vector $x_i \in \mathbb{R}^n$, and $\Psi \in \mathbb{R}^{nL}$ is an overcomplete sparsifying dictionary ($\Psi =$

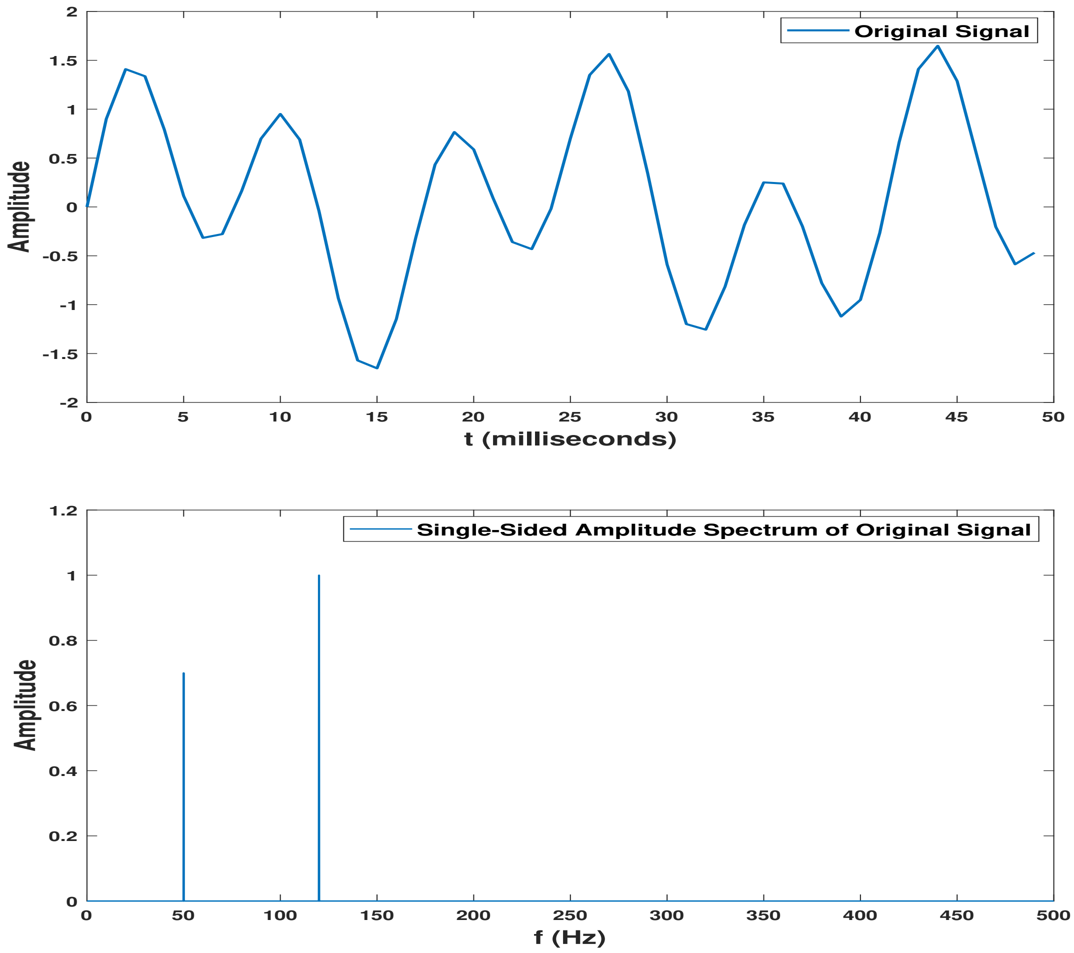


Figure 3.2: The original signal and its single-sided amplitude spectrum.

$[\psi_1|\psi_2|...|\psi_L]$, $\psi_i \in \mathbb{R}^n$, $n < L$), $S \in \mathbb{R}^{Lp}$ contains p sparse vectors $s_i \in \mathbb{R}^L$ corresponding to the patches in X. In topics related to sparsity and image denoising, the term *dictionary* is referred to *matrix*, and every column of a dictionary is called *atom*. Also in (3.2), $\|.\|_F$ stands for the Frobenius norm which equals the square root of the sum of the squares of its elements, and $\|.\|_0$ is the number of nonzero coefficients of s_i, and T_0 is an positive integer number and equals to the order of sparsity.

Equation (3.2) is an optimization problem with two objectives: finding an overcomplete dictionary Ψ, and sparse representation matrix S which every column of this matrix should have at most T_0 nonzero elements. The noisy image patches X are the only available infor-

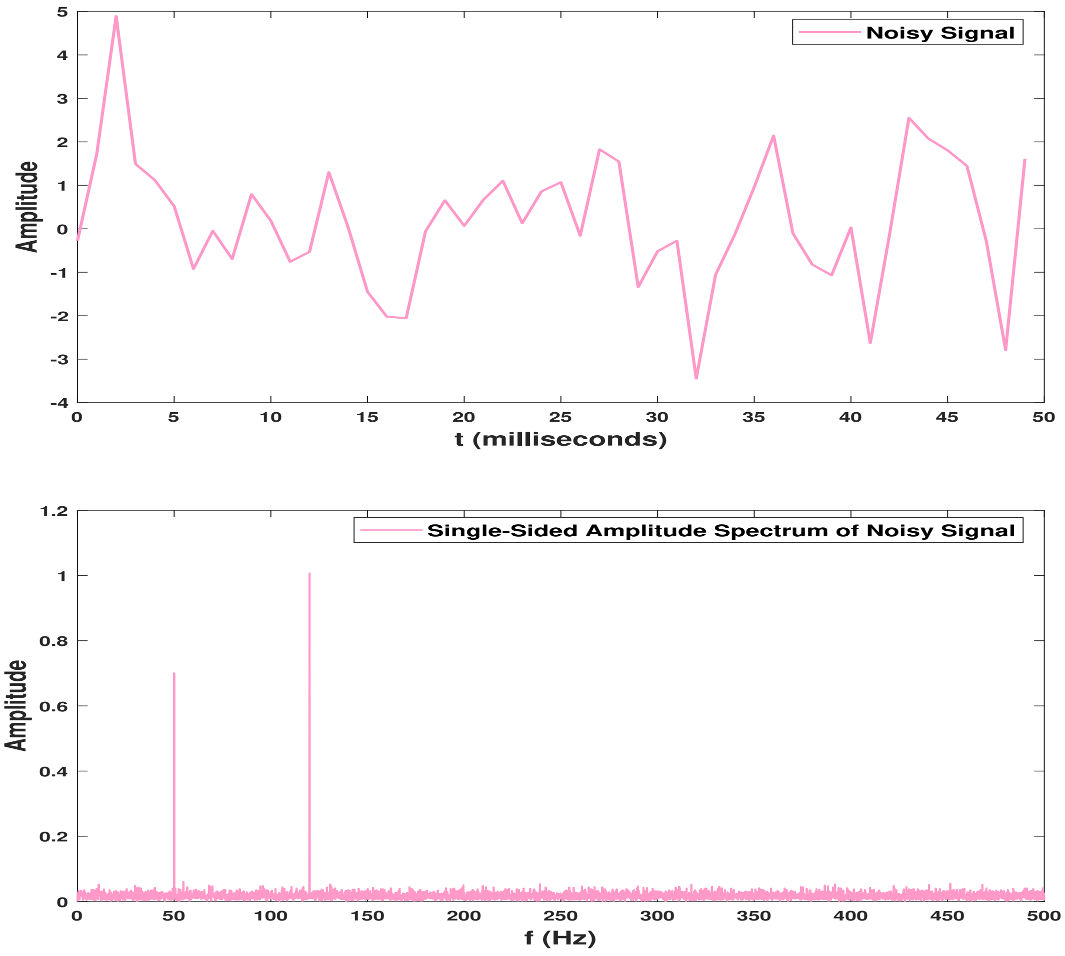


Figure 3.3: The noisy signal and its single-sided amplitude spectrum.

mation to solve this problem. A common approach to solve this problem is breaking it into two separate optimization problems[91]:

- Sparse coding: finding the S by assuming Ψ as a fixed known dictionary.
- *Dictionary update:* finding the Ψ by assuming S as a fixed, known matrix.

The initial value for the dictionary can be chosen in different ways, such as a random matrix, part of the image patches, or designed based on prior knowledge. The algorithm iterates between these two steps until there is no significant update in dictionary elements.

3.3.2 Sparse Coding

Every noisy patch $x_i \in \mathbb{R}^n$ can be represented as a linear combination of atoms $\Psi = [\psi_1|\psi_2|...|\psi_L]\,, \psi_i \in \mathbb{R}^n,\ n < L$, i.e. $x_i = \Psi s_i = \sum_{j=1}^{j=L} \alpha_j \psi_j$ where $s_i = (\alpha_1, \alpha_2, ..., \alpha_L)$. In the sparse coding step, we suppose the dictionary is known. Ψ is an overcomplete dictionary and yields an underdetermined system of equations where the number of unknowns is more than the number of equations. Due to the rank deficiency of given Ψ, there are many s that can be transferred to the same x. However, by adding a sparsity constraint on the s, a unique sparse vector can be obtained or approximated. In this case, s is called a sparse representation of x. With this regard, the first step can be done by solving the following optimization problem:

$$\min \|s_i\|_0 \text{ s.t } \|x_i - \Psi s_i\|_2^2 \leq \epsilon \quad i = 1, 2, ..., p. \tag{3.3}$$

where ϵ is a small number. There are numerous algorithms [111, 112, 113, 114, 115, 116] to solve (3.3), the orthogonal matching pursuit (OMP) [116] is a well-known greedy algorithm with the order of complexity $O(T_0 nL)$. This algorithm is mostly used in image denoising problems. After running the OMP algorithm for p times, sparse vectors $\{s_i\}_{i=1}^{i=p}$ correspond to noisy image patches $\{x_i\}_{i=1}^{i=p}$ are obtained.

3.3.3 Dictionary Update

In this step, the algorithm assumes X and S as fixed inputs and updates the dictionary Ψ by minimizing the following error.

$$\|E\|_F^2 = \|X - \Psi S\|_F^2 \tag{3.4}$$

There are different methods for updating a dictionary, but one of the best methods to achieve minimum error is based on the method of optimal direction [91]. In this approach, we take the derivative of (2) with respect to the dictionary Ψ, and by zeroing the derivative output, optimal dictionary based on given X and S can be obtained as follows.

$$\begin{aligned} \frac{\partial \|E\|_F^2}{\partial \Psi} &= 0 \\ (X - \Psi S)S^T &= 0 \\ \Psi_{opt} &= (XS^T)(SS^T)^{-1} \end{aligned} \tag{3.5}$$

Equation (3.5) gives the best local minimum error. After obtaining an updated Ψ, step one is repeated to generate an updated sparse matrix. These two steps continue until there are no considerable changes in the updating steps (depending on the given stop threshold). For example after t iterations, the t^{th} Ψ_{opt} and S are used to generate denoised image patches, $X_{denoised} = \Psi_{opt}^{t^{\text{th}}} S^{t^{\text{th}}}$, and by knowing the position of each patch and taking average of pixels, we can recover the denoised image.

Implementing the two steps mentioned above requires solid computational resources. For p noisy image patches and executing two steps for t iterations, we need to run $t \times p$ times an OMP algorithm and t times (3.5). In real applications, the number of patches is huge and requires powerful resources. One solution to handle this problem is outsourcing the denoising task to the cloud. We propose lightweight encryption over noisy patches that enables the cloud to implement the perfect denoising and preserve the privacy of data simultaneously.

3.4 Lossless Privacy-Preserving Image Denoising

This section proposes a novel approach to preserve the privacy of users' data and the quality of image denoising tasks. As discussed3.3.1, the required data that should be sent to the cloud is the noisy image patches X and initial dictionary Ψ. We should encrypt the noisy image patches to preserve users' privacy. Then, we use some parts of image patches as an initial dictionary.

We use two random matrices so as to encrypt noisy image patches. For every noisy image, we randomly permute the position of the patches with a random permutation matrix $U \in \mathbb{R}^{p \times p}$. Then we multiply an orthonormal random matrix $Q \in \mathbb{R}^{n \times n}$ (e.g. Gaussian random matrix with independent and identically distributed random variables) by permuted X as follows:

$$\begin{cases} X^{\star} &= XU \\ \acute{X} &= QX^{\star} = QXU \end{cases} \tag{3.6}$$

where $X^{\star}$ is the permuted noisy patches matrix, and $\acute{X}$ is the encrypted noisy image patches that are sent to the cloud for the image denoising task.

Then upon a user request of the denoising task, the cloud solves (3.2) with the encrypted input data as follows:

$$\min_{\acute{\Psi},\acute{S}} \|\acute{X} - \acute{\Psi}\acute{S}\|_F^2 \quad \text{s.t} \quad \forall_i \|\acute{s}_i\|_0 \leq T_0 \tag{3.7}$$

where $\acute{S}$ is the encrypted sparse matrix, and i^{th} column of this matrix is $\acute{s}_i$. And $\acute{\Psi} = Q\Psi$ is the encrypted sparsifying dictionary. T3.3.2, the cloud breaks (3.7) into two steps: sparse coding and dictionary update to recover the denoised patches.

3.4.1 Proof of Lossless Property

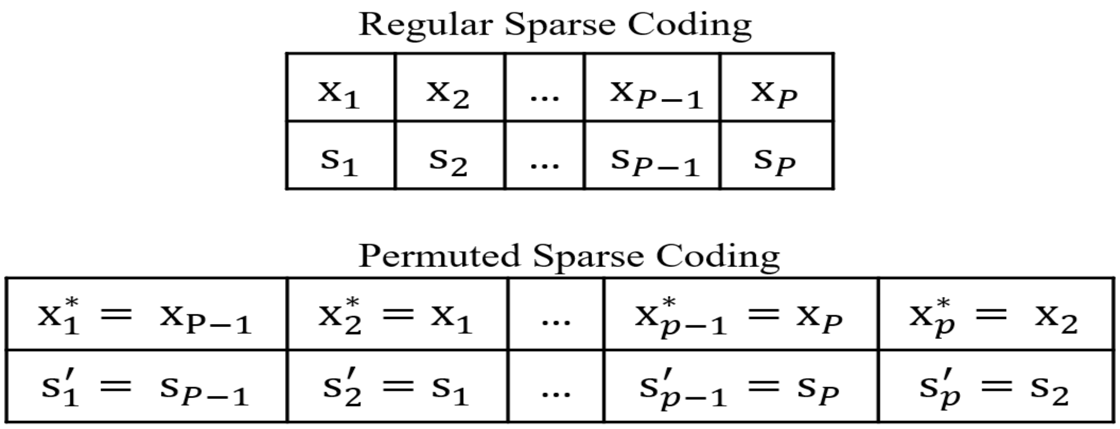
We compare step by step solving the ordinary denoising (3.2) with that of the (3.7). In the first step which is sparse coding, the cloud solves the following problem

$$\min \|\acute{s}_i\|_0 \ \text{ s.t } \ \|\acute{x}_i - \acute{\Psi}\acute{s}_i\|_2^2 \leq \epsilon \quad i = 1, 2, ..., p \tag{3.8}$$

where $\acute{x}_i$ is the i^{th} column of the noisy image patches $\acute{X}$. In (3.8), there is a norm-2 (Euclidean norm) condition. When we multiply an orthonormal matrix with a vector, the norm-2 of that vector does not change. We substitute corresponding values of $\acute{x}_i = Qx_i^\star$ and $\acute{\Psi} = Q\Psi$ in (3.8), and use the norm-2 property to simplify it as follows.

$$\begin{cases} \min \|\acute{s}_i\|_0 \;\; \text{s.t} \;\; \|Qx_i^\star - Q\Psi\acute{s}_i\|_2^2 \le \epsilon \quad i = 1, 2, ..., p \\ \|Qx_i^\star - Q\Psi\acute{s}_i\|_2^2 = \|Q(x_i^\star - \Psi\acute{s}_i)\|_2^2 = \|x_i^\star - \Psi\acute{s}_i\|_2^2 \\ \Rightarrow \min \|\acute{s}_i\|_0 \;\; \text{s.t} \;\; \|x_i^\star - \Psi\acute{s}_i\|_2^2 \le \epsilon \quad i = 1, 2, ..., p \end{cases} \tag{3.9}$$

The final minimization problem in (3.9) gives the same result as (3.3). Notice that $X^\star$ contains the same noisy image patches with different orders that do not affect the sparse coding step. Hence, after this step, we would have the permuted sparse representation $\acute{S} = SU$ corresponding to the $X^\star$. In Fig. 3.4, we illustrate an example of permuted sparse coding. It is evident that after recovery, the results are the same; however, the order is not the same as the regular case.



Regular Sparse Coding

x_1	x_2	...	x_{P-1}	x_P
s_1	s_2	...	s_{P-1}	s_P

Permuted Sparse Coding

$x_1^* = x_{P-1}$	$x_2^* = x_1$	...	$x_{p-1}^* = x_P$	$x_p^* = x_2$
$s_1' = s_{P-1}$	$s_2' = s_1$	...	$s_{p-1}' = s_P$	$s_p' = s_2$

Figure 3.4: An example of showing the effect of permutation on the sparse coding step.

In the next step, we need to implement dictionary update by using the encrypted noisy patches $\acute{X}$ and permuted sparse matrix $\acute{S}$ obtained from the last step. As we discussed in section 3.3.3, we suppose $\acute{X}$ and $\acute{S}$ as fixed inputs, then we minimize the error $\|\acute{X} - \acute{\Psi}\acute{S}\|_F^2$ by taking derivative with respect to the dictionary $\acute{\Psi}$, and by zeroing the derivative output, optimal dictionary based on given $\acute{X}$ and $\acute{S}$ can be obtained as follows.

$$\acute{\Psi}_{opt} = (\acute{X}\acute{S}^T)(\acute{S}\acute{S}^T)^{-1} \tag{3.10}$$

By substituting $\acute{X} = QXU$ and $\acute{S} = SU$ in (3.10), and regarding the orthonormal property of the permutation matrix, we can achieve a direct relationship between the regular sparsifying dictionary and encrypted one as follows.

$$\begin{cases} \acute{\Psi}_{opt} = (QXUU^TS^T)(SUU^TS^T)^{-1} \\ \acute{\Psi}_{opt} = (QXS^T)(SS^T)^{-1} \\ \acute{\Psi}_{opt} = Q\Psi_{opt} \end{cases} \tag{3.11}$$

After running t times the first and second steps, when there is no further update on dictionary elements, the algorithm will be stopped, and the t^{th} encrypted sparse matrix $\acute{S}$ and sparsifying dictionary $\acute{\Psi}_{opt}$ will be used to generate encrypted denoised patches as follows.

$$\begin{cases} \acute{X}_{denoised} = \acute{\Psi}^{t^{\text{th}}}_{opt}\acute{S}^{t^{\text{th}}} \\ \acute{X}_{denoised} = (Q\Psi^{t^{\text{th}}}_{opt})(S^{t^{\text{th}}}U) = QX_{denoised}U \end{cases} \tag{3.12}$$

The service provider in the cloud generates $\acute{X}_{denoised}$ which is an encrypted (permuted and perturbed) version of ordinary denoised patches. The real user by having the real keys Q and U, and calling the decryption function ($Dec(.)$) can decrypt the cloud's output with minimal complexity as follows.

$$Dec(\acute{X}_{denoised}) = Q^T\acute{X}_{denoised}U^T = X_{denoised} \tag{3.13}$$

Then, by knowing the position of each patch and taking the average of pixels, the real user can obtain the denoised image. Therefore our proposed method can perfectly maintain the quality of denoising. In the next section, we analyze the robustness of the proposed method, and we show how this method completely perturbs the statistical and visual information of the noisy image.

3.4.2 Security Analysis

One challenging attack in security designs is called known-plaintext attack (KPA) [117, 118]. In this type of attack, the attacker or curious cloud has access to the plaintext and corresponding ciphertext. The attacker tries to guess the private keys in order to decrypt the future ciphertexts. In our proposed method, the user generates new private keys for every denoising request. In other words, for every noisy image, the user generates two new and independent random matrices so as to prevent potential KPA attacks. Since the keys are updated in each session, the attacker cannot use the previous keys to decrypt the future ciphertexts.

We also consider a scenario in which an attacker tries to implement a brute force attack to obtain the keys: random permutation matrix $U \in \mathbb{R}^{p \times p}$ and an orthonormal random matrix $Q \in \mathbb{R}^{n \times n}$. There is $p!$ (! represents the factorial operation) possible permutation matrices $\in \mathbb{R}^{p \times p}$. Similar to every cryptosystem, the length of the key(s) or the extent of search space plays an important role in defining the level of secrecy of that cryptosystem. In patched-based image denoising problems, p, the number of noisy image patches is large. Given an $r \times r$ pixels image, there are $p = (r - \sqrt{n} + 1)^2$ possible $\sqrt{n} \times \sqrt{n}$ pixels patches. For example, for a small image of size 32× 32 pixels, and patch size 8× 8 pixels, the number of possible patches is p = (32-8+1)^2 = 625, and the number of possible permutation matrices is 625! which is a huge number. It is very hard to calculate this number even with the powerful resources available on the cloud. In practice, the size of p is much larger than 625, for example in [91] and [92], p = 11000 and p = 50000 patches have been used, respectively.

In addition, if we suppose that the attacker estimates the permutation matrix, she still needs to guess the perturbation matrix Q. Since we update the Q in every session, it is similar to a one-time random projection method. Authors in [119] theoretically proved the strength of this type of encryption against different attack scenarios. They reached this point that if in every session a new and large enough non-Gaussian random matrix Q to be used, only the energy of each patch would be leaked. However, if we use a Gaussian random matrix, there would not be any leak of information. A direct consequence of their result is the fact that the attacker would not obtain any information from the given ciphertext if we use a random Gaussian matrix.

We further evaluated the strength of our method against the side-channel attack [120]. In this type of attack, the attacker (a curious cloud or a man-in-the-middle) measures the difference between execution times in order to estimate the keys. Since we only encrypt the data, our method does not impose any extra task on the cloud side. In other words, the cloud executes the same procedure if the noisy image is sent. One important consequence of this property is having close processing times which can nullify a side-channel attack.

We verified this claim by running a systematic experiment to compare the execution times. In Fig. 3.5, we compare the execution times of the denoising process with and without proposed encryption. In this experiment, the *Baseline* and *Proposed* refer to the denoising process with the noisy data (plaintext) and encrypted noisy data (ciphertext), respectively. To have reliable results, we repeated the experiment 100 times, and each time we used new random keys to encrypt the data. We implemented all processes in the MATLAB environment on a system with specifications of 16G RAM and processor *Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz*. The size of the noisy image was 64×64 pixels. We set the size of the sliding window to 8×8 pixels and generated 3249 noisy patches. The results show that the execution time of the proposed approach closely follows the baseline.

In Table 3.1, we present the average execution times on client and cloud-side. On the cloud-side, the average execution times of the proposed method (49.65 seconds) and baseline (49.48 seconds) are very close. Hence, the attacker cannot implement the side-channel attack based on measuring the execution times. Besides, in Table 3.1, the average execution times in client-side (encryption plus decryption) is much lower than that of the cloud-side. It verifies

Table 3.1: The average execution times (seconds) of running denoising algorithm with noisy images (baseline) and encrypted noisy images (proposed) on the cloud-side.

Cloud-Side Denoising	
Proposed	Baseline
49.65	49.48
Client-Side: Encryption plus Decryption	
0.56	

the high complexity of the denoising process and the reason to outsource the denoising task to the cloud.

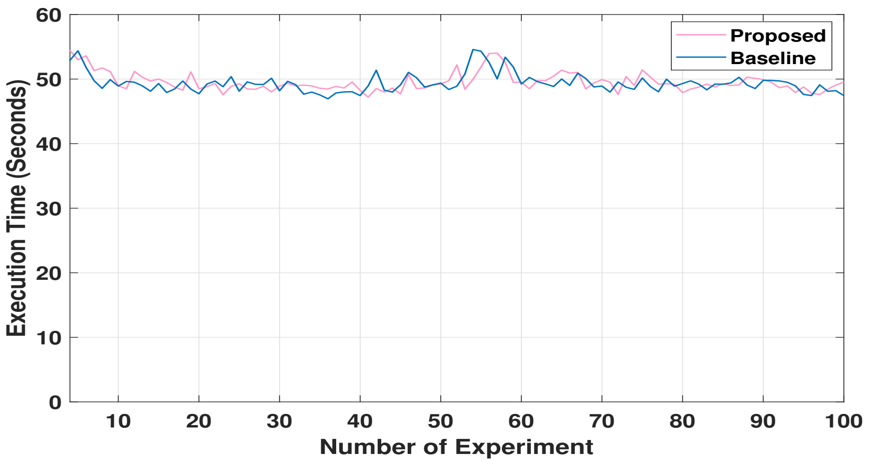


Figure 3.5: Comparing the execution times of running denoising task on noisy patches (baseline) and encrypted noisy patches on the cloud-side (proposed).

3.5 Experimental Results

In this section, we compare the proposed method to the baseline to verify the lossless property. In addition, we compare our method with the previous works to show its superiority in maintaining the quality of denoising. In this study, we use five images *'House, Boat, Barbara, Lena, Fruits'* that are mostly used for evaluating the quality of image denoising techniques. The size of the original images were 256×256 pixels, and to generate the noisy images and accordingly noisy patches, we used white additive Gaussian noise with two levels of noise, $\sigma = 15$ and $\sigma = 20$. Also, we used a Gaussian random matrix with independent and identically distributed random variables to generate the perturbation key.

We used two quantitative measures: peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) to evaluate the performance of this method compared to the baseline. PSNR measures the ratio between the maximum possible power of the input image and the power of corrupting noise that affects the representation of the image. Also, SSIM is a perceptual metric that takes both original and noisy images to evaluate the image quality degradation. It ranges $0<\text{SSIM}<1$, higher SSIM means a higher quality of denoising. Table 3.2 shows the comparison results. The baseline refers to the case that we do not encrypt the data, i.e., implementing ordinary denoising on the plaintext. Then, we encrypted the noisy patches and re-ran the same denoising algorithm on this encrypted data. After doing denoising, we decrypted the image, and as presented in Table 3.2, the quality of the proposed method is exactly the same as the quality of the baseline. This experiment verifies our claim about the lossless property of this method.

Furthermore, we compared the average loss of previous works with the proposed method in $\sigma = 20$. As presented in Table 3.3, previous works cannot maintain the quality of image denoising. For the brevity and since we proposed a solid theoretical proof for the lossless property, we omitted the results for other σ values and images. We released our MATLAB code publicly for reproducibility and future comparisons[3].

In addition, Fig. 3.6 illustrates the effect of perturbation on the cloud. The first column from the left corresponds to the noisy images with $\sigma = 20$, the second column corresponds to the denoised images based on the ordinary method (baseline), the third column corresponds to the encrypted images that have been denoised on the cloud, yet not decrypted. And the fourth column shows the decrypted image by the real user. This experiment shows that while the cloud does the denoising task, it will have a completely noise-like image. Therefore, the cloud would not be able to observe the content of images. Also, the real user can decrypt and

[3]https://github.com/hadizand/privacy-preserving-image-denoising.git

Table 3.2: Comparing the quality of proposed privacy-preserving image denoising outsourcing with the regular outsourcing (baseline).

Image	Method	$\sigma = 15$		$\sigma = 20$	
		PSNR	SSIM	PSNR	SSIM
House	Noisy	24.64	0.44	22.16	0.34
	Baseline	30.46	0.69	27.94	0.58
	Proposed	30.46	0.69	27.94	0.58
Boat	Noisy	24.70	0.58	22.20	0.48
	Baseline	29.76	0.80	27.68	0.71
	Proposed	29.76	0.80	27.68	0.71
Barbara	Noisy	24.64	0.61	22.19	0.45
	Baseline	30.56	0.84	28.42	0.76
	Proposed	30.56	0.84	28.42	0.76
Lena	Noisy	24.63	0.52	22.15	0.42
	Baseline	30.55	0.78	28.17	0.68
	Proposed	30.55	0.78	28.17	0.68
Peppers	Noisy	24.82	0.59	22.37	0.49
	Baseline	30.52	0.83	28.43	0.75
	Proposed	30.52	0.83	28.43	0.75

Table 3.3: Comparing the average loss of proposed method with previous works.

	Loss in PSNR	Loss in SSIM
LSH-voting [107]	0.74	0.10
External DB [107]	0.52	0.08
Denoising in the dark [109]	0.25	0
Proposed	0	0

access the denoised image by having the keys. As shown in Fig. 3.6, our proposed method does not affect the visual features of the denoising task.

3.6 Conclusion

In this study, we proposed a novel lossless privacy-preserving image denoising method that protects the private data within noisy images and maintains the quality of the denoising task concurrently. In this method, we used two random keys to permute and perturb the noisy image to transform the noisy patches into an encrypted signal. Cloud implemented the denoising task on this encrypted signal. After denoising, the output signal was still encrypted, and the real user would be able to decrypt the denoised image by knowing the keys. We evaluated the security of this method against known plaintext and brute-force attacks. Also, we theoretically proved the lossless property of this method. We also verified the applicability of this approach by implementing our experiments on five real images, and two well-known evaluation metrics, PSNR and SSIM, were used to compare our results with the baseline. Our results show that this method can maintain the quality of denoising while imposing limited complexity to the client and cloud sides.

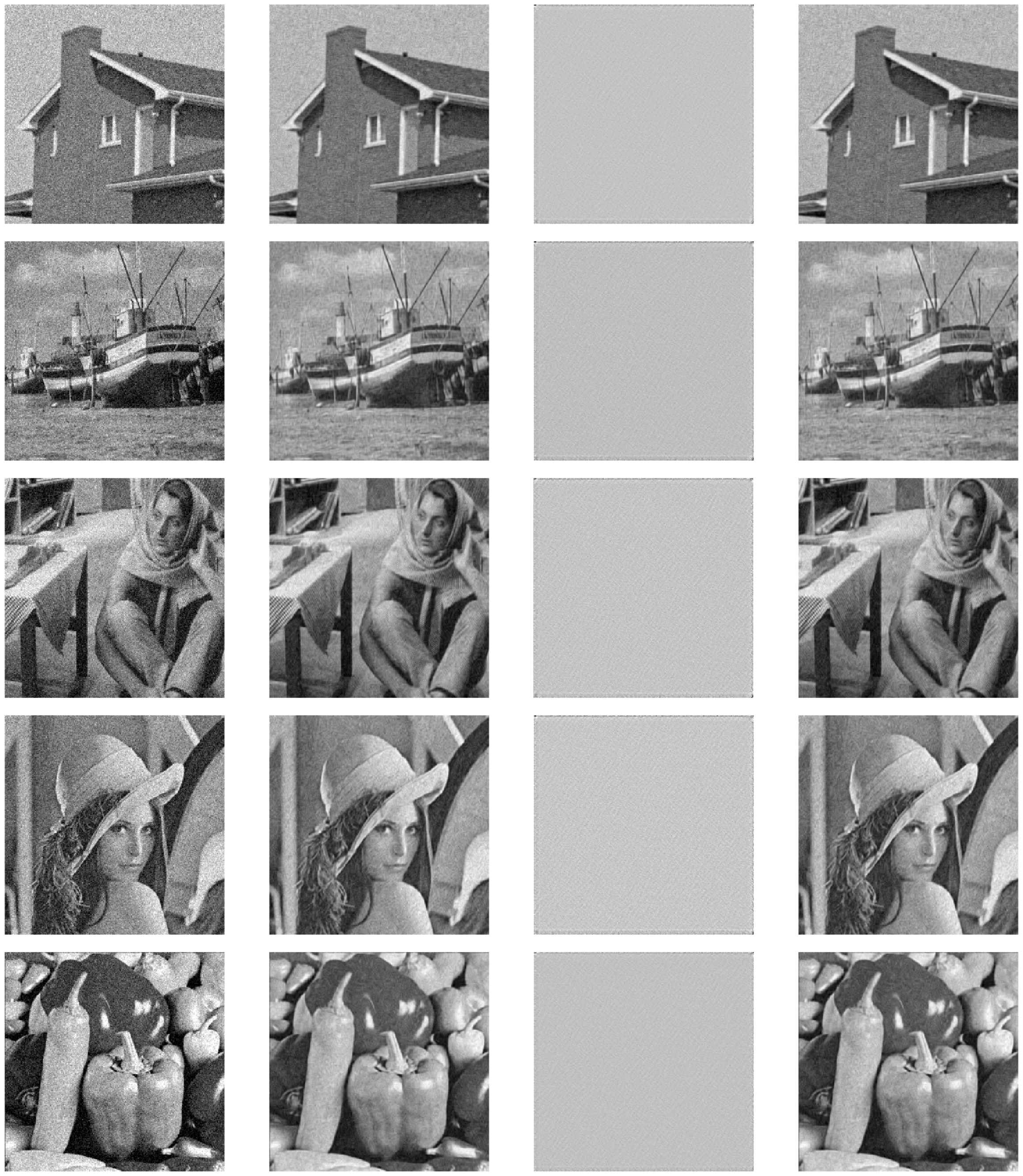


Figure 3.6: Experiment results of visual information exposed in the cloud-side, and comparing the quality of denoised image with the baseline.

References

[1] Hadi Zanddizari, Behnam Zeinali, and J. Morris Chang. Generating black-box adversarial examples in sparse domain. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–10, 2021.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[3] R. A. Novoa J. Ko S. M. Swetter H. M. Blau A. Esteva, B. Kuprel and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[4] Hadi Zanddizari, Nam Nguyen, Behnam Zeinali, and J. Morris Chang. A new preprocessing approach to improve the performance of cnn-based skin lesion classification. *Med. Biol. Eng. Comput.*, 59:1123–1131, 2021.

[5] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, 2016.

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[7] F. Marra, D. Gragnaniello, and L. Verdoliva. On the vulnerability of deep learning to adversarial attacks for camera model identification. *Signal Processing: Image Communication*, 65:240–248, 2018.

[8] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, page 16–25, New York, NY, USA, 2006. Association for Computing Machinery.

[9] A. D. Joseph M. Barreno, B. Nelson and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[10] Chin-Feng Yu and Hsing-Kuo Pao. Virtual adversarial active learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5323–5331, 2020.

[11] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, 2014.

[12] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning, 2016.

[13] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey, 2018.

[14] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[16] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan L. Yuille. Adversarial attacks beyond the image space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4297–4306, 2019.

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[18] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, 2016.

[19] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2017.

[20] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, 2016.

[21] Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, 2017.

[22] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Generic black-box end-to-end attack against state of the art api call based malware classifiers, 2018.

[23] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis, 2016.

[24] Chuan Guo, Jared S. Frank, and Kilian Q. Weinberger. Low frequency adversarial perturbation, 2019.

[25] Yash Sharma, Gavin Weiguang Ding, and Marcus Brubaker. On the effectiveness of low frequency perturbations, 2019.

[26] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning, 2017.

[27] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[28] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems, 2017.

[29] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[30] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[31] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

[32] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[35] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[36] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[37] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. *Object Recognition with Gradient-Based Learning*, pages 319–345. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[38] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, Rujun Long, and Patrick McDaniel. Technical report on the cleverhans v2.1.0 adversarial examples library, 2018.

[39] https://www.kaggle.com/alessiocorrado99/animals10.

[40] G. Jocher et al. ultralytics/yolov5: v3.1 - bug fixes and performance improvements, 2020.

[41] https://github.com/hadizand/las-adversarial-example.git.

[42] B. Zeinali, A. Ayatollahi, and M. Kakooei. A novel method of applying directional filter bank (dfb) for finger-knuckle-print (fkp) recognition. In *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*, pages 500–504, 2014.

[43] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

[44] David Gutman, Noel CF Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (isbi) 2016, hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1605.01397*, 2016.

[45] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5:180161, 2018.

[46] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172. IEEE, 2018.

[47] Marc Combalia, Noel CF Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Allan C Halpern, Susana Puig, and Josep Malvehy. Bcn20000: Dermoscopic lesions in the wild. *arXiv preprint arXiv:1908.02288*, 2019.

[48] Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H Maier-Hein. Brain tumor segmentation and radiomics survival prediction: Contribution to the brats 2017 challenge. In *International MICCAI Brainlesion Workshop*, pages 287–297. Springer, 2017.

[49] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[50] HP Ng, SH Ong, KWC Foong, PS Goh, and WL Nowinski. Medical image segmentation using k-means clustering and improved watershed algorithm. In *2006 IEEE southwest symposium on image analysis and interpretation*, pages 61–65. IEEE, 2006.

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[52] Lequan Yu, Hao Chen, Qi Dou, Jing Qin, and Pheng-Ann Heng. Automated melanoma recognition in dermoscopy images via very deep residual networks. *IEEE transactions on medical imaging*, 36(4):994–1004, 2016.

[53] Lei Bi, Jinman Kim, Euijoon Ahn, Ashnil Kumar, Michael Fulham, and Dagan Feng. Dermoscopic image segmentation via multistage fully convolutional networks. *IEEE Transactions on Biomedical Engineering*, 64(9):2065–2074, 2017.

[54] Yading Yuan, Ming Chao, and Yeh-Chi Lo. Automatic skin lesion segmentation using deep fully convolutional networks with jaccard distance. *IEEE transactions on medical imaging*, 36(9):1876–1886, 2017.

[55] Manu Goyal, Moi Hoon Yap, and Saeed Hassanpour. Multi-class semantic segmentation of skin lesions via fully convolutional networks. *arXiv preprint arXiv:1711.10449*, 2017.

[56] Sulaiman Vesal, Shreyas Malakarjun Patil, Nishant Ravikumar, and Andreas K Maier. A multi-task framework for skin lesion detection and segmentation. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 285–293. Springer, 2018.

[57] Amira Soudani and Walid Barhoumi. An image-based segmentation recommender using crowdsourcing and transfer learning for skin lesion extraction. *Expert Systems with Applications*, 118:400–410, 2019.

[58] Jun Zhang and Jinglu Hu. Image segmentation based on 2d otsu method with histogram analysis. In *2008 International Conference on Computer Science and Software Engineering*, volume 6, pages 105–108. IEEE, 2008.

[59] Juliana M Haggerty, Xiao N Wang, Anne Dickinson, Chris J O'Malley, and Elaine B Martin. Segmentation of epidermal tissue with histopathological damage in images of haematoxylin and eosin stained human skin. *BMC medical imaging*, 14(1):7, 2014.

[60] Ch Hima Bindu and K Satya Prasad. An efficient medical image segmentation using conventional otsu method. *International Journal of Advanced Science and Technology*, 38(1):67–74, 2012.

[61] J Premaladha and KS Ravichandran. Novel approaches for diagnosing melanoma skin lesions through supervised and deep learning algorithms. *Journal of medical systems*, 40(4):96, 2016.

[62] E. Buza, A. Akagic, and S. Omanovic. Skin detection based on image color segmentation with histogram and k-means clustering. In *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, pages 1181–1186, 2017.

[63] Kevin McGuinness and Noel E. O'Connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434 – 444, 2010. Interactive Imaging and Vision.

[64] Matt Berseth. Isic 2017-skin lesion analysis towards melanoma detection. *arXiv preprint arXiv:1703.00523*, 2017.

[65] Tianyi Zhao, Dashan Gao, Jiao Wang, and Zhaozheng Tin. Lung segmentation in ct images using a fully convolutional neural network with multi-instance and conditional adversary loss. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 505–509. IEEE, 2018.

[66] Xiao Xiao, Shen Lian, Zhiming Luo, and Shaozi Li. Weighted res-unet for high-quality retina vessel segmentation. In *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pages 327–331. IEEE, 2018.

[67] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *IEEE transactions on medical imaging*, 37(12):2663–2674, 2018.

[68] Nabil Ibtehaz and M Sohel Rahman. Multiresunet: Rethinking the u-net architecture for multimodal biomedical image segmentation. *Neural Networks*, 121:74–87, 2020.

[69] Mohammad H Jafari, Nader Karimi, Ebrahim Nasr-Esfahani, Shadrokh Samavi, S Mohamad R Soroushmehr, K Ward, and Kayvan Najarian. Skin lesion segmentation in clinical images using deep learning. In *2016 23rd International conference on pattern recognition (ICPR)*, pages 337–342. IEEE, 2016.

[70] Jeremy Kawahara and Ghassan Hamarneh. Multi-resolution-tract cnn with hybrid pretrained and skin-lesion trained layers. In *International workshop on machine learning in medical imaging*, pages 164–171. Springer, 2016.

[71] Tanzila Saba, Muhammad Attique Khan, Amjad Rehman, and Souad Larabi Marie-Sainte. Region extraction and classification of skin cancer: A heterogeneous framework of deep cnn features fusion and reduction. *Journal of medical systems*, 43(9):289, 2019.

[72] Amirreza Mahbod, Gerald Schaefer, Chunliang Wang, Rupert Ecker, Georg Dorffner, and Isabella Ellinger. Investigating and exploiting image resolution for transfer learning-based skin lesion classification, 2020.

[73] K. M. Hosny, M. A. Kassem, and M. M. Foaud. Skin cancer classification using deep learning and transfer learning. In *2018 9th Cairo International Biomedical Engineering Conference (CIBEC)*, pages 90–93, 2018.

[74] A. A. Adegun and S. Viriri. Deep learning-based system for automatic melanoma detection. *IEEE Access*, 8:7160–7172, 2020.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[76] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[77] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[78] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017.

[79] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[80] François Chollet et al. Keras, 2015. https://keras.io, [accessed April. 1, 2020].

[81] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404, 1999.

[82] Alexia Briassouli, Jenny Benois-Pineau, and Alexander Hauptmann. *Overview of Multimedia in Healthcare*, pages 1–6. Springer International Publishing, Cham, 2015.

[83] F. Hu, D. Xie, and S. Shen. On the application of the internet of things in the field of medical and health care. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 2053–2058, 2013.

[84] Y. Zhang, H. Huang, Y. Xiang, L. Y. Zhang, and X. He. Harnessing the hybrid cloud for secure big image data service. *IEEE Internet of Things Journal*, 4(5):1380–1388, 2017.

[85] Xun Yi, Fang-Yu Rao, Elisa Bertino, and Athman Bouguettaya. Privacy-preserving association rule mining in cloud computing. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, page 439–450, New York, NY, USA, 2015. Association for Computing Machinery.

[86] L. Liu, L. Mou, X. X. Zhu, and M. Mandal. Skin lesion segmentation based on improved u-net. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–4, 2019.

[87] S. S. Han, G. H. Park, W. Lim, M. S. Kim, J. I. Na, I. Park, and S. E. Chang. Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network. *PLoS One*, 13(1), 2018.

[88] J. Rajan, B. Jeurissen, J. Sijbers, and K. Kannan. Denoising magnetic resonance images using fourth order complex diffusion. In *2009 13th International Machine Vision and Image Processing Conference*, pages 123–127, 2009.

[89] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[90] M. Scetbon, M. Elad, and P. Milanfar. Deep k-svd denoising. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[91] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, Nov 2006.

[92] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, Dec 2006.

[93] K. Zhang, W. Zuo, and L. Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[94] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017.

[95] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5882–5891, 2017.

[96] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. In *Advances in Neural Information Processing Systems*, pages 1680–1689, 2018.

[97] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[98] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486, 2011.

[99] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3d transform-domain collaborative filtering. In *2007 15th European Signal Processing Conference*, pages 145–149, 2007.

[100] Y. Xie, S. Gu, Y. Liu, W. Zuo, W. Zhang, and L. Zhang. Weighted schatten p -norm minimization for image denoising and background subtraction. *IEEE Transactions on Image Processing*, 25(10):4842–4857, 2016.

[101] Alex Bateman and Matt Wood. Cloud computing. *Bioinformatics*, 25(12):1475–1475, 05 2009.

[102] S. Wang, K. Gu, S. Ma, W. Lin, X. Liu, and W. Gao. Guided image contrast enhancement based on retrieved images in cloud. *IEEE Transactions on Multimedia*, 18(2):219–232, 2016.

[103] Zahir Tari, Xun Yi, Uthpala S. Premarathne, Peter Bertok, and Ibrahim Khalil. Security and privacy in cloud computing: Vision, trends, and challenges. *IEEE Cloud Computing*, 2(2):30–38, 2015.

[104] Patrick Eugster, Seema Kumar, Savvas Savvides, and Julian James Stephen. Ensuring confidentiality in the cloud of things. *IEEE Pervasive Computing*, 18(1):10–18, 2019.

[105] E. Bertino. Zero trust architecture: Does it help? *IEEE Security & Privacy*, 19(05):95–96, sep 2021.

[106] Maha Tebaa, Said El Hajji, and Abdellatif El Ghazi. Homomorphic encryption method applied to cloud computing. In *2012 National Days of Network Security and Systems*, pages 86–89, 2012.

[107] Y. Zheng, H. Duan, X. Tang, C. Wang, and J. Zhou. Denoising in the dark: Privacy-preserving deep neural network based image denoising. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2019.

[108] Y. Zheng, C. Wang, and J. Zhou. Toward secure image denoising: A machine learning based realization. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6936–6940, 2018.

[109] Y. Zheng, H. Cui, C. Wang, and J. Zhou. Privacy-preserving image denoising from external cloud databases. *IEEE Transactions on Information Forensics and Security*, 12(6):1285–1298, 2017.

[110] X. Hu, W. Zhang, K. Li, H. Hu, and N. Yu. Secure nonlocal denoising in outsourced images. *ACM Transactions on Multimedia Computing, Communications and Applications*, 12(3):40:1–40:23, 2016.

[111] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, jan 2001.

[112] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, may 2009.

[113] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean-Luc Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58(2):1094–1121, feb 2012.

[114] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, 9(3):317–334, jun 2008.

[115] Dror Baron, Shriram Sarvotham, and Richard G. Baraniuk. Bayesian compressive sensing via belief propagation. *IEEE Transactions on Signal Processing*, 58(1):269–280, jan 2010.

[116] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*. IEEE Comput. Soc. Press.

[117] Bo Zhang, Di Xiao, Mengdi Wang, and Jia Liang. Privacy-preserving compressed sensing for image simultaneous compression-encryption applications. In *2021 Data Compression Conference (DCC)*, pages 283–292, 2021.

[118] J. Mohamedmoideen Kader Mastan and R. Pandian. Cryptanalytic attacks on a chaos-based image encrypting cryptosystem. In *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pages 1049–1052, 2021.

[119] Tiziano Bianchi, Valerio Bioglio, and Enrico Magli. Analysis of one-time random projections for privacy preserving compressed sensing. *IEEE Transactions on Information Forensics and Security*, 11(2):313–327, feb 2016.

[120] Zelalem Birhanu Aweke and Todd Austin. Øzone: Efficient execution with zero timing leakage for modern microarchitectures. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 153–153, 2017.

Appendix A: Copyright Permissions

The permission below is for the use of the content in Chapter 1.

CCC RightsLink®

Home | Help | Live Chat | Hadi Zanddizari

IEEE Requesting permission to reuse content from an IEEE publication

Generating Black-Box Adversarial Examples in Sparse Domain

Author: Hadi Zanddizari

Publication: IEEE Transactions on Emerging Topics in Computational Intelligence

Publisher: IEEE

Date: Dec 31, 1969

Copyright © 1969, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

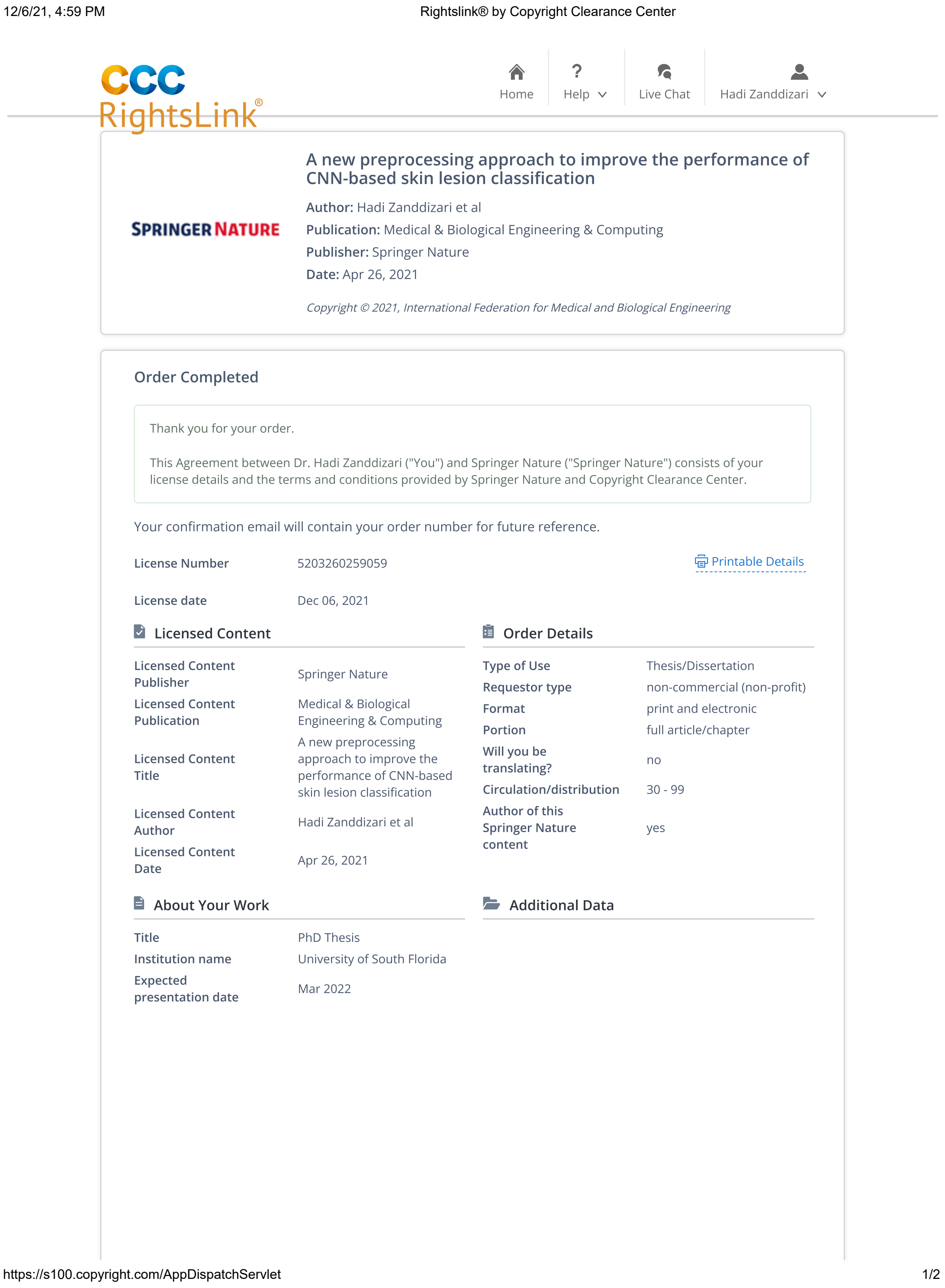
Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

The permission below is for the use of the content in Chapter 2.



12/6/21, 4:59 PM Rightslink® by Copyright Clearance Center

Home | Help | Live Chat | Hadi Zanddizari

A new preprocessing approach to improve the performance of CNN-based skin lesion classification

Author: Hadi Zanddizari et al

Publication: Medical & Biological Engineering & Computing

Publisher: Springer Nature

Date: Apr 26, 2021

Copyright © 2021, International Federation for Medical and Biological Engineering

Order Completed

Thank you for your order.

This Agreement between Dr. Hadi Zanddizari ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

License Number	5203260259059
License date	Dec 06, 2021

Printable Details

Licensed Content

Licensed Content Publisher	Springer Nature
Licensed Content Publication	Medical & Biological Engineering & Computing
Licensed Content Title	A new preprocessing approach to improve the performance of CNN-based skin lesion classification
Licensed Content Author	Hadi Zanddizari et al
Licensed Content Date	Apr 26, 2021

Order Details

Type of Use	Thesis/Dissertation
Requestor type	non-commercial (non-profit)
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	30 - 99
Author of this Springer Nature content	yes

About Your Work

Title	PhD Thesis
Institution name	University of South Florida
Expected presentation date	Mar 2022

Additional Data

https://s100.copyright.com/AppDispatchServlet 1/2

About the Author

Hadi Zanddizari is a Ph.D. majoring in Electrical Engineering at the University of South Florida. He received a BSc in Communication Engineering, and a Master's in Electrical Engineering from the Iran University of Science and Technology, Tehran, Iran, in 2015. From 2015 to 2017 he worked at Pardis Technology Park as a researcher and programmer. His research interests include deep learning, object detection, semantic segmentation, cybersecurity, and data privacy.