Generating Black-Box Adversarial Examples in Sparse Domain

Hadi Zanddizari, *Student Member, IEEE*, Behnam Zeinali, *Student Member, IEEE*, and J. Morris Chang, *Senior Member, IEEE*

***Abstract*—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 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 paper, we propose a novel approach to generate a black-box attack in sparse domain whereas the most important information of an image can be observed. Our investigation shows that large sparse (LaS) components play a critical role in the performance of image classifiers. Under this presumption, to generate 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 fooling image classifiers with a fewer number of 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 YOLOv5 model as an object detection algorithm. Mean squared error (MSE) and peak signal to noise ratio (PSNR) are used as quality metrics. We also present a theoretical proof to connect these metrics to the level of perturbation in the sparse domain.**

***Index Terms*—Convolutional neural network, black-box attack, deep learning, sparse representation.**

I. 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. [1]–[3]. While almost CNNs perform well and provide high accuracy, their robustness toward some malicious attacks still are not acceptable [4]–[6]. Applying some perturbation on the input data may totally 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 [7]–[11]. By slightly perturbing the input data, 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 [5], [12], [13]. 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, but still we expect the classifiers 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

$$\min ||\mathbf{r}||_2 \quad \text{s.t.} \quad C(\mathbf{x}+\mathbf{r}) \neq C(\mathbf{x}) \tag{1}$$

where $\mathbf{r}$ is adversarial perturbation, $||.||_2$ is the Euclidean norm or ℓ_2 norm, $\mathbf{x}$ is the legitimate image (original image), and $C(.)$ yields the classifier's output label. Based on (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 a 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'. In targeted misclassification, the attacker tries to fool the classifier to yield a targeted label. For example, the same legitimate image with a label '1'is labeled as a specific number like '8'by the classifier. In this study, 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's 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) [14], beyond the image space approach that uses physical space features of 3D images, [15], deepfool [16],

Manuscript received 18 June 2021; revised 31 August 2021; accepted 24 September 2021. Date of publication 4 November 2021; date of current version 22 July 2022.

The authors are with the Department of Electrical Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: hadiz@usf.edu; behnamz@usf.edu; chang5@usf.edu).

Digital Object Identifier 10.1109/TETCI.2021.3122467

Jacobean-based Saliency Map Attack (JSMA) [4]. 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:

$$\mathbf{x}^* = \mathbf{x} + \epsilon \, sign\left(\nabla_{\mathbf{x}} \mathcal{J}\left(\mathbf{x}, c\right)\right) \tag{2}$$

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 and parameters, and training dataset [17]–[20]. This type of attack is more practical because in many cases having access to the training dataset is not possible. 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 [12].

In a non-adaptive black-box attack, an attacker can have access only to the distribution of the training dataset [21]. 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 training distribution of the dataset and also 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 [12], [22]. Although these types of systems may provide 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 assumes 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 [26], authors proposed generating adversarial examples based on perturbing one-pixel of an image through differential evolution. Although this method could fool almost CNN models due to the inherent features of differential evolution, there is no limit for the number of queries to attack the model. Papernot *et al.* [17] 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 [18]. 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 dataset for training the local model. Recently, Hosseini *et al.* [23] 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 they add some perturbations to the input data, and based on some threshold and probability functions, they assign the label '$Null$' to the perturbed image. Then, they retrain the model with clean and new adversarial examples which have 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.

There are some black-box approaches that 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 of them 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 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 II-A, we show the difference between LaS and LoF components.

Focusing on LaS components have been used in many image processing and compression techniques. The JPEG codec [27] 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 [27]. Intuitively, image classifiers are mostly consider specific components which bear more information of an image. We verify this property of image classifiers by implementing systematic experiments (Section II-B). 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 are evaluated. Our experiment results

show that the proposed approach can fool the classifiers with less number of 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 number of 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 study, the summary of our contributions are as follow:

- We introduce a black-box approach to generate adversarial examples in the sparse domain in order 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 very recent black-box attacks which focused on LoF components, we show that the LaS components can fool the classifiers with a fewer number of queries.
- We proposed an analytical approach to show the relation between the perturbation level in the sparse domain and its effect on the pixel domain. Our results show the proposed method decreases the number of required queries to fool the ML models and increases the misclassification rate of ML models.

II. Sparsity

Sparsity has been widely used in many applications such as image denoising, deblurring, super resolution, and compression [28]–[32]. An image signal $\mathbf{X} \in \mathbb{R}^{p\times q}$ can be reshaped to a vector $\mathbf{x} \in \mathbb{R}^{N=p\times q}$ where N is the number of pixels. Dictionary $\mathbf{D} \in \mathbb{R}^{N\times L}$ is a matrix which linear combination of its columns $\mathbf{d}_i$ can approximately represent the $\boldsymbol{x}$ as follow

$$\mathbf{x} = \sum_{i\in\{1,2,..L\}} s_i \mathbf{d}_i = \mathbf{D}\mathbf{s} \tag{3}$$

where $\mathbf{s} \in \mathbb{R}^L$ is the weight vector. If $\mathbf{D}$ provides a weight vector with only k large and $l-k$ negligible or zero elements, then $\mathbf{D}$ and $\mathbf{s}$ can be called as a sparsifying dictionary and sparse representation of input $\mathbf{x}$, 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 ($\mathbf{D} \in \mathbb{R}^{N\times N}$ *and* $||\mathbf{d_i}||_2 = 1$). The coefficients of DCT dictionary can be obtained as follows,

$$d_{i,j} = a_{i,j} \cos \frac{\pi(2i-1)(j-1)}{2N} \quad i,j \in 1,2,\ldots,N$$

$$a_{i,j} = \begin{cases} \sqrt{\frac{1}{N}} & j = 1 \\ \sqrt{\frac{2}{N}} & j \neq 1 \end{cases} \tag{4}$$

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 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. It is evident that reconstructed images based on only 30%, 20%, or 10% of its LaS components can still preserve lots of visual information of the image.

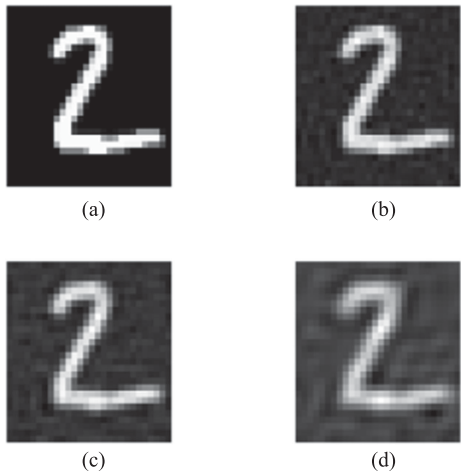


Fig. 1. Transferring image into the sparse domain and zeroing small elements of sparse signal: (a) original image, (b) zeroing 70%, (c) 80%, and (d) 90% of small elements.

A. Difference Between LaS and LoF Components

Sparse domain enables us to have access to 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 belong to any frequency bands. 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 level of intersection between LaS and LoF components, we used 10,000 color images of size 256×256 pixels. The images had three color channels, and we mapped each channel into the sparse domain, separately. Then we selected $N = (k \times k \times 3)$ 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. Fig. 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 does 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 critical information of an image to only its low frequency space.

TABLE I
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 [25] & [26]**	**30% of LoF [25] & [26]**	**50% of HiF [26]**	**30% of HiF [26]**
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

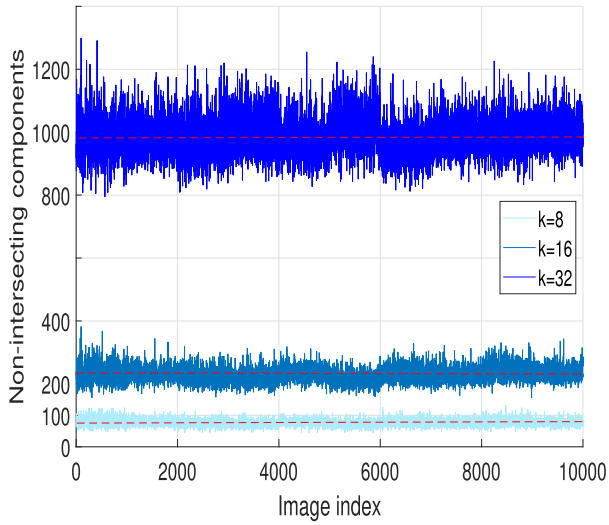


Fig. 2. The number of non-intersecting components of each image.

In next section, we evaluate the effects of manipulating different frequency bands on the performance of CNN models.

B. *Effect of LaS Components on CNN Models*

Sparse transformation enables us to compact the energy of the signal 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 manipulating LaS, LoF, or HiF components can affect classifiers' performance?". In this study, we empirically show 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 [33], ResNet50 [34], InceptionV3 [35], MobileNets [36], and DenseNet121 [37]. We used CIFAR-10 dataset which is a color and balanced image dataset with complex background. This dataset contains 50,000 training samples and 10,000 test samples belong to 10 classes. We trained these models with 50,000 training samples, and then we input the original 10,000 test samples (without any changes or manipulation) to obtain the ground truth accuracy of each trained model (Table I). In next step, via DCT dictionary we transferred all 10,000 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 back each image to the pixel domain, and input them to 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 I, the accuracies belong to LaS components test datasets are much closer to their corresponding ground truth accuracies. While keeping only LoF or HiF components lead to considerable lost 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 which is one of the best image classifiers that has been introduced by Google in 2019, has the 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 on, only 50% of LaS components affect classifiers, the other 50% components does not much affect the accuracy. This experiment helps us to find out which frequency components mostly affect the CNN models. By having this information, we would be able to add perturbation on important components in order to fool image classifiers. Also this experiment verified the results of [25] that showed the importance of LoF vs HiF components. They reached to this conclusion that perturbing LoF components is more effective than perturbing HiF components. For the 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 next section, we add a limited perturbation to LaS and LoF components, to see which of them can fool the classifiers in a fewer number of queries.

III. PERTURBING LAS COMPONENTS

In the adaptive black-box attack there is no prior information about the model's parameters and distribution of the training dataset, yet attacker can query the label of legitimate sample and corresponding perturbed sample. However, if the number of query to be increased, the system may identify a malicious activity. Obviously, an adversarial attack is more practical if it fools classifiers in a fewer number of queries. we designed a systematic experiment to evaluate the effectiveness of adding perturbation on LaS components. Our results demonstrate that proposed approach requires fewer number of 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 50,000 training samples of CIFAR 10 dataset. We used 10,000 test samples of CIFAR-10 dataset that had never been used in training

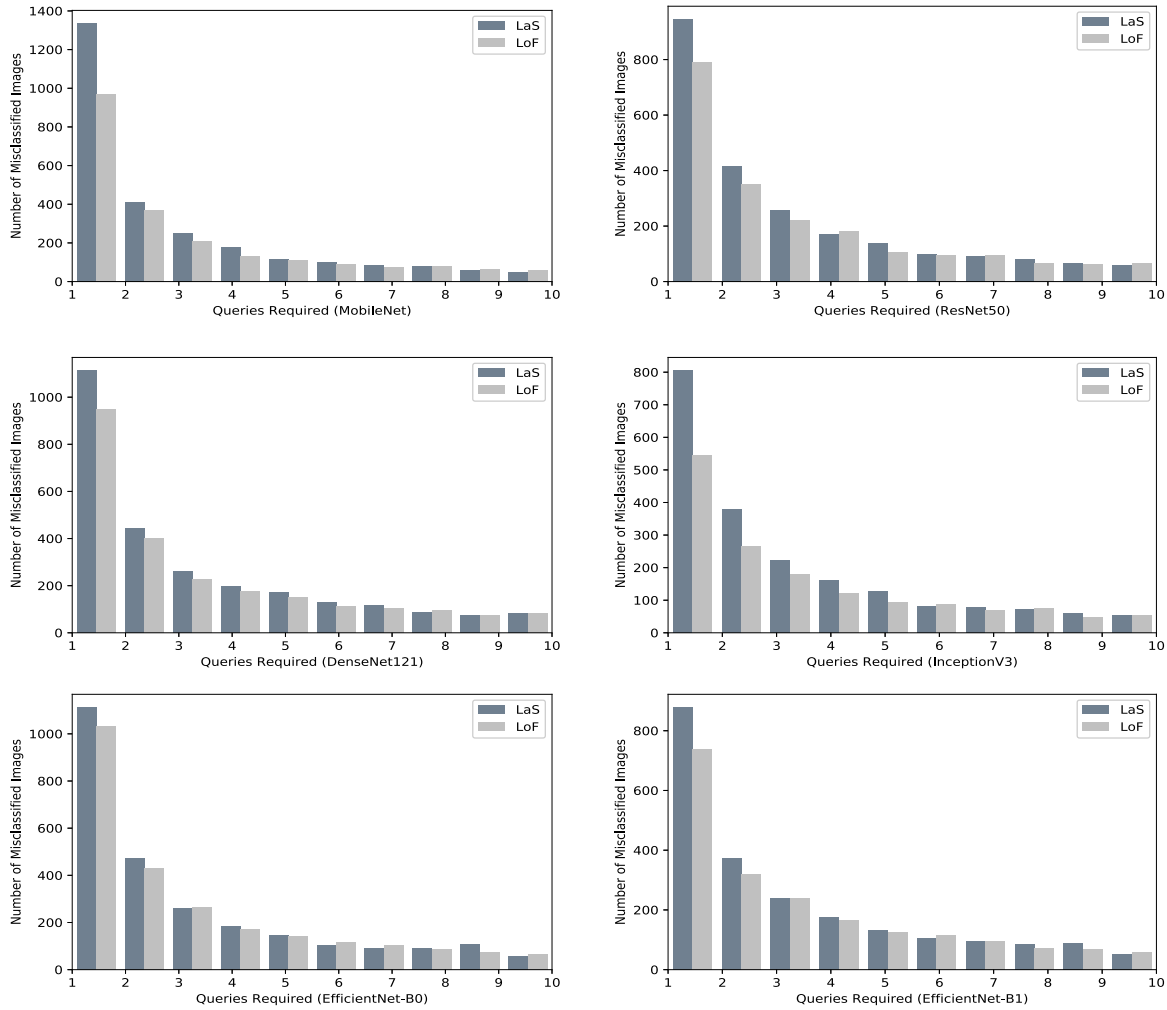


Fig. 3. Comparing the required number of queries to fool CNN models based on proposed approach (LaS), and LoF [25].

process to apply the attacks. We utilized 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 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, the histograms of required number of queries to successfully fool aforementioned CNN models are demonstrated. The distributions of successful attacks show that manipulating LaS components can fool the CNN models in a fewer number of queries. Fig. 4 shows the number of all misclassified images in query less or equal to 10. In this experiment, we firstly 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 to the experiment (because it was already misclassified). Hence, the number of misclassified images in Figs. 3 and 4 are only due to the perturbation on samples.

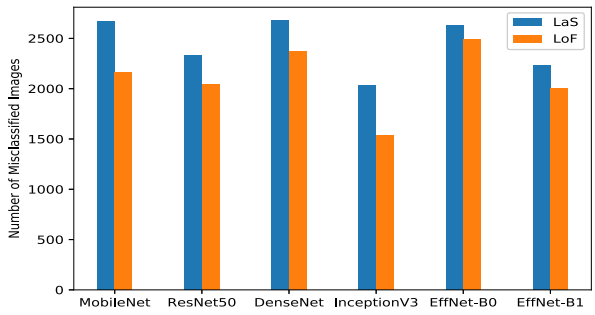


Fig. 4. Comparing number of misclassified samples for query less or equal to 10 based on LaS and LoF [25].

IV. Case Study: Directed Perturbation

In this section, we propose a method for adding noise to the LaS components in order 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 training stage. Then, the attacker can verify or find the input sample's label by observing the output of the objective model. In this section, we assume the attacker can have multiple samples of each class and their labels. Suppose the available dataset is $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{i=p}$ which contains p samples and each sample belongs to one class out of m available classes, i.e., $C(\mathbf{x}_i) \in \{c_j\}_{j=1}^{j=m}$. We map all samples

of the dataset into the sparse domain via DCT dictionary $\mathbf{D}$. Doing so, $\mathbf{S} = \{\mathbf{s}_i\}_{i=1}^{i=p}$ would be obtained where $\mathbf{s}_i$ is the sparse representation of the $\mathbf{x}_i$. 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

$$\hat{S} = \{\hat{\mathbf{s}}_i\}_{i=1}^{i=p}, ||\hat{\mathbf{s}}_i||_0 = k, ||\hat{\mathbf{s}}_i||_2 = 1 \tag{5}$$

where $||.||_0$ is the zero-norm of a vector which counts the number of non-zero elements of a vector. Sparse vector $\hat{\mathbf{s}}_i$ contains information of the positions and normalized values of the k largest elements of $\mathbf{s}_i$ which belong to class $C(s_i)$. Then for a given $(\hat{\mathbf{s}}_i,\ C(s_i))$, we find the most correlated sparse vector $(\hat{\mathbf{s}}_j,\ C(s_j) \neq C(s_i))$. In other words, sparse vector $\hat{\mathbf{s}}_j$ is the closest sparse vector to the $\hat{\mathbf{s}}_i$, but they belong to different classes. We used the inner product of two vectors $\langle \hat{\mathbf{s}}_i, \hat{\mathbf{s}}_j \rangle$ to calculate the correlation. If we change the k most important elements of $\hat{\boldsymbol{s}}_i$ with respect to the k most important elements of $\hat{\mathbf{s}}_j$, some information and features of $\hat{\mathbf{s}}_j$ can be transferred into the $\hat{\mathbf{s}}_i$. If some nonzero elements of $\hat{\mathbf{s}}_i$ and $\hat{\boldsymbol{s}}_j$ 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 $\mathbf{d}_{ij}$ as follows:

$$\mathbf{d}_{ij} = \hat{\mathbf{s}}_i - \hat{\mathbf{s}}_j \tag{6}$$

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

$$\widetilde{\mathbf{s}}_i = \mathbf{s}_i - \delta \mathbf{d}_{ij} \tag{7}$$

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

$$\widetilde{\mathbf{x}}_i = \mathbf{D}\widetilde{\mathbf{s}}_i \tag{8}$$

where $\widetilde{\mathbf{x}}_i$ is the adversarial example. Since the response of ML classifier for $\mathbf{s}_j$ is $C(s_j)$, when we add the elements of $\hat{\mathbf{s}}_j$ to the $\hat{\mathbf{s}}_i$, 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.

$$MSE = \frac{||\mathbf{x}_i - \widetilde{\mathbf{x}}_i||_2^2}{N} \tag{9}$$

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

where h is the maximum fluctuation in the input image data type. For example, since we normalized all image dataset to the range of [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 $||\mathbf{x}_i - \widetilde{\mathbf{x}}_i||_2^2 = ||\delta \mathbf{D}\mathbf{d}_{\mathbf{ij}}||_2^2$ and due to the fact that the dictionary $\mathbf{D}$ is an orthonormal dictionary and the δ is a scalar value, $||\mathbf{x}_i - \widetilde{\mathbf{x}}_i||_2^2 = \delta^2 ||\mathbf{d}_{\mathbf{ij}}||_2^2$. (9) can be further simplified to obtain more straightforward relation between δ and MSE or $PSNR$ in pixel domain as follows:

$$MSE = \frac{\delta^2}{N} ||\mathbf{d}_{\mathbf{ij}}||_2^2 = \frac{\delta^2}{N} ||\hat{\mathbf{s}}_i - \hat{\mathbf{s}}_j||_2^2 = \frac{2\delta^2}{N} (1 - \langle \hat{\mathbf{s}}_i, \hat{\mathbf{s}}_j \rangle) \tag{11}$$

where $\langle \cdot, \cdot \rangle$ is the inner product operation of two vectors. Since both $\hat{\mathbf{s}}_i$ and $\hat{\mathbf{s}}_j$ are normalized vectors, their inner product equals a number belongs to $[-1, 1]$. Hence MSE can be bounded $0 \leq MSE \leq \frac{4\delta^2}{N}$. However, as we choose two most correlated sparse vectors, their inner product is usually greater than zero. Hence, the upper bound of MSE may be smaller, i.e. $0 \leq MSE \leq \frac{2\delta^2}{N}$. 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 has its effect on the inner product.

TABLE II
COMPARING MISCLASSIFICATION RATES OF DIRECTED ATTACK OVER SIX CNN MODELS BASED ON PROPOSED METHOD (LAS) AND RECENT METHOD (LOF) [25]

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

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 $MSE \leq 0.001$. Table II 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 [38] with 60,000 training samples of MNIST dataset and achieved the accuracy of 98.2% which means 1.8% misclassification rate over 10,000 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 corresponding perturbed test dataset. Then we input all these 36 adversarial sets to the LeNet classifier to observe the response of the network. Fig. 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

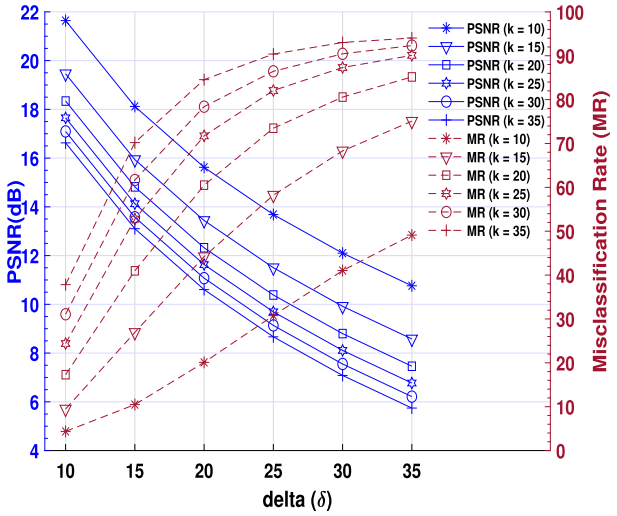


Fig. 5. Generating adversarial examples with different level of perturbation on LeNet.

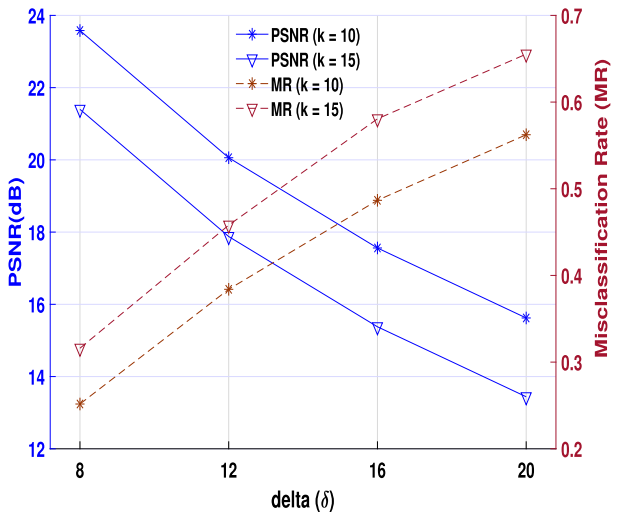


Fig. 6. Generating adversarial examples with different level of perturbation on SVM classifier.

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 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. Fig. 6 shows that the SVM classifier is highly vulnerable to the proposed attack.

We compared our approach with a recent work by Papernot *et al.* [17] which is not based on frequency domain. We used the Cleverhans library [39], 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

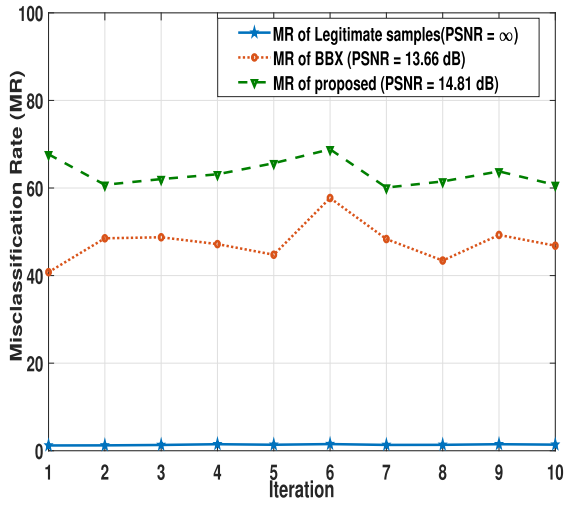


Fig. 7. Comparing the misclassification rate of proposed method of perturbation and recent practical black-box (BBX) approach. [17].

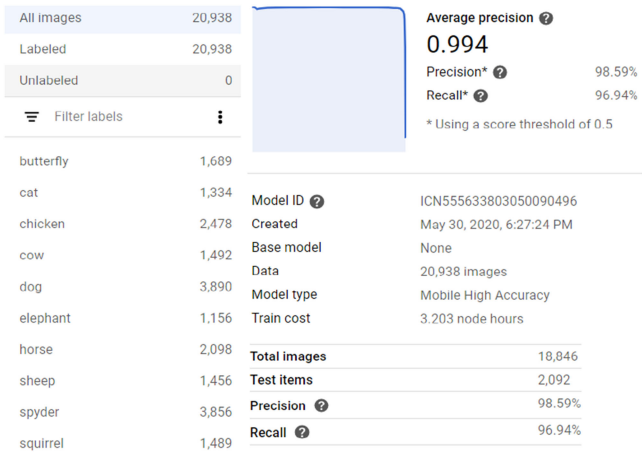


Fig. 8. Information of dataset and trained model by Google Cloud Vision.

trained model on both adversarial sets was recorded. Fig. 7 shows for $\delta = 15$ and $k = 20$, our proposed adversarial examples have higher misclassification rate than that of the previous work, while our method has a higher PSNR which means less perceptible perturbation.

V. 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 [40]. We used a high-resolution dataset which contained 20938 samples belong to 10 animals "spider, dog, cat, squirrel, sheep, butterfly, horse, elephant, cow, chicken" [41]. Fig. 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 samples out of 500 samples were misclassified. Also, adding noise to LoF components led to 129

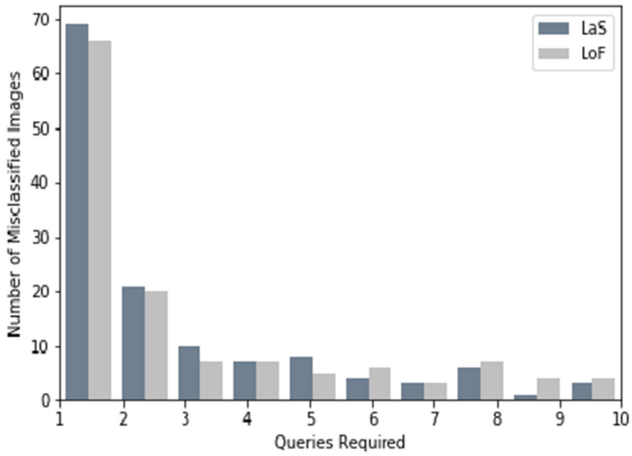


Fig. 9. Comparing the required number of queries to fool a TFlite model trained by GoogleAPI based on proposed approach (LaS), and LoF [25].

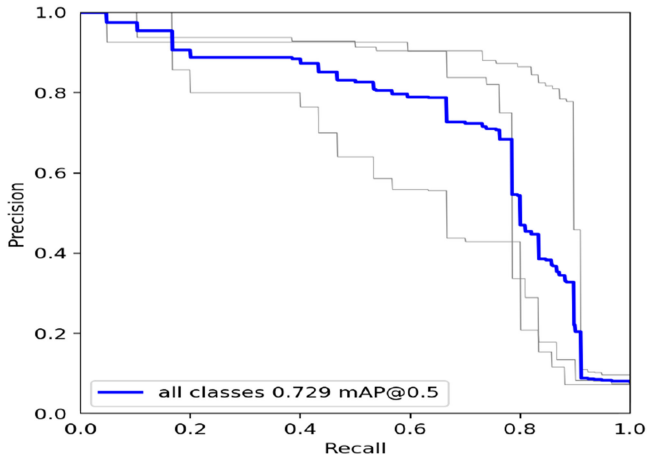


Fig. 11. Performance of YOLOv5 over skin lesion dataset (ISIC-2017).

Fig. 10. Samples of attacking Google Cloud Vision. The most left column corresponds to the legitimate image, and the other three columns are misclassified adversarial examples. we supposed $MSE > 0.001$ as a failure and corresponding adversarial examples are bounded by red color boxes. We supposed $MSE = 0.001$ as a success and corresponding examples are enclosed by a green color box.

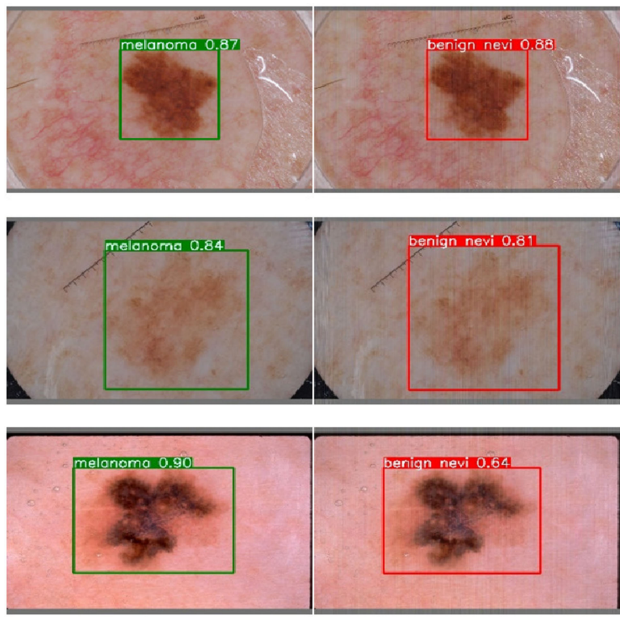


Fig. 12. Samples of attacking the object detection algorithm (YOLOv5). The left column corresponds to the legitimate images that have been correctly detected and classified, and the right column corresponds to the misclassified objects.

misclassified samples. Fig. 9 shows the number of required queries to fool the TFlite model based on both methods. In addition, Fig. 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 which 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 [42]. 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. In certain sensitive applications, if the model fails to detect the object correctly or predict the label wrongly, it may cause irreversible consequences. 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 belong to three skin lesion classes: *melanoma, nevus, and seborrheic keratosis*. We resized the input samples into 640×640 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. Fig. 11 shows the performance of trained model over test dataset. Precision is a metric that measures how accurate is the predictions, while recall measures how good the model finds all the positive cases. IoU measures the overlap between predicted box around the object with the ground truth. The model achieved mean Average Precision (mAP) equal to 0.72 over three classes. In next step, we randomly selected some test samples that had never been used in 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. 12, we only showed few adversarial examples that had been misclassified. However, there were adversarial samples that model 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, trained object detection model, and the annotation files of ISIC-2017 dataset publicly for reproducibility [43].

VI. Conclusion

In this work, we proposed a new approach for generating adversarial examples in the sparse domain. We show LaS components are different 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 build 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 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 it 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 much fewer number of queries than that of 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 aforementioned models. By introducing the potential threat within this type of attack, an appropriate defense mechanism can be investigated in the future. Moreover, we used 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.

References

[1] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[2] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, Jan. 2017.

[3] H. Zanddizari, N. Nguyen, B. Zeinali, and J. M. Chang, "A new preprocessing approach to improve the performance of CNN-based skin lesion classification," *Med. Biol. Eng. Comput.*, vol. 59, pp. 1123–1131, May 2021.

[4] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS P)*, 2016, pp. 372–387.

[5] C. Szegedy *et al.*, "Intriguing properties of neural networks," Dec. 2013, *arXiv:1312.6199*.

[6] F. Marra, D. Gragnaniello, and L. Verdoliva, "On the vulnerability of deep learning to adversarial attacks for camera model identification," *Signal Process., Image Commun.*, vol. 65, pp. 240–248, Jul. 2018.

[7] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proc. ACM Symp. Inf., Comput. Commun. Secur., Ser*, New York, NY, USA: ACM, 2006, pp. 16–25.

[8] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, Nov. 2010.

[9] B. Miller *et al.*, "Adversarial active learning," in *Proc. Workshop Artif. Intell. Secur. Workshop, Ser.* New York, NY, USA: ACM, 2014, pp. 3–14.

[10] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 984–996, Apr. 2014.

[11] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," Nov. 2016, *arXiv:1611.03814*.

[12] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," Sep. 2018, *arXiv:1810.00069*.

[13] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," Jan. 2018, *arXiv:1801.00553*.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," Dec. 2014, *arXiv:1412.6572*.

[15] X. Zeng *et al.*, "Adversarial attacks beyond the image space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4297–4306.

[16] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," Nov. 2015, *arXiv:1511.04599*.

[17] N. Papernot *et al.*, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur., Ser.*, New York, NY, USA: ACM, 2017, pp. 506–519.

[18] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," May 2016, *arXiv:1605.07277*.

[19] N. Narodytska and S. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1310–1318.

[20] I. Rosenberg, A. Shabtai, L. Rokach, and Y. Elovici, "Generic black-box end-to-end attack against state of the art API call based malware classifiers," Jul. 2017, *arXiv:1707.05970*.

[21] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *Proc. 25th USENIX Conf. Secur. Symp., Ser. USA, USENIX Assoc.*, 2016, pp. 601–618.

[22] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618.

[23] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems," Mar. 2017, *arXiv:1703.04318*.

[24] C. Guo, J. S. Frank, and K. Q. Weinberger, "Low frequency adversarial perturbation," in *Proc. The 35th Uncertainty Artif. Intell. Conf.*, 2020, pp. 1127–1137.

[25] Y. Sharma, G. W. Ding, and M. A. Brubaker,"On the effectiveness of low frequency perturbations," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3389–3396.

[26] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[27] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no.1, pp. xviii–xxxiv, Feb. 1992

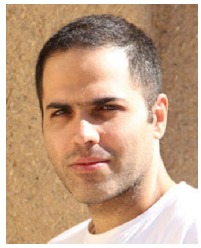
[28] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[29] S. Ujan, S. Ghorshi, M. Pourebrahim, and S. A. Khoshnevis, "On the use of compressive sensing for image enhancement," in *Proc. UKSim-AMSS 18th Int. Conf. Comput. Model. Simul.*, 2016, pp. 167–171.

[30] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.

[31] J. Zepeda, C. Guillemot, and E. Kijak, "Image compression using sparse representations and the iteration-tuned and aligned dictionary," *IEEE J. Sel. Top. Signal Process.*, vol. 5, no. 5, pp. 1061–1073, Sep. 2011.
[32] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, Jul. 2007.
[33] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.
[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015, *arXiv:1512.00567*.
[36] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[37] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016, *arXiv:1608.06993*.
[38] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*. London, U.K.: Springer, 1999, pp. 319–345.
[39] N. Papernot *et al.*, "Technical report on the CleverHans v2.1.0 adversarial examples library," 2018, *arXiv:1610.00768*.
[40] Google, "Deploy machine learning models on mobile and IoT devices," [Online]. Available: https://www.tensorflow.org/lite
[41] [Online]. Available: https://www.kaggle.com/alessiocorrado99/animals10
[42] G. Jocher *et al.*, "ultralytics/yolov5: V3.1 - Bug fixes and performance improvements," Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370
[43] [Online]. Available: https://github.com/hadizand/LaS-Adversarial-example.git

Hadi Zanddizari (Student Member, IEEE) is currently a Research Assistant with the Department of Electrical Engineering, University of South Florida, Tampa, FL, USA. His research interests include deep learning, object detection, semantic segmentation, cybersecurity, and data privacy.



Behnam Zeinali (Student Member, IEEE) received the M.Sc. degree in electrical engineering from the Iran University of Science and Technology, Tehran, Iran, in 2013. He is currently working toward the Ph.D. degree with the University of South Florida, Tampa, FL, USA. From 2013 to 2019, he was with the industry, as a Programmer, Researcher, and Developer in the field of AI. His research interests include machine and deep learning, computer vision, and mobile application programming.

J. Morris Chang (Senior Member, IEEE) received the Ph.D. degree from North Carolina State University, Raleigh, NC, USA. He is currently a Professor with the Department of Electrical Engineering, University of South Florida, Tampa, FL, USA. His past industrial experiences include positions with the Texas Instruments, Microelectronic Center of North Carolina, and AT&T Bell Labs. His research interests include cyber security and data privacy, machine learning, and mobile computing. He was the recipient of the University Excellence in Teaching Award at Illinois Institute of Technology in 1999. He was inducted into the NC State University ECE Alumni Hall of Fame in 2019. He is a Handling Editor of the *Journal of Microprocessors and Microsystems* and the Editor of the IEEE IT PROFESSIONAL.