On the Importance of Encrypting Deep Features

Xingyang Ni
Tampere University
Tampere, Finland
xingyang.ni@tuni.fi

Heikki Huttunen
Visy Oy
Tampere, Finland
heikki.huttunen@visy.fi

Esa Rahtu
Tampere University
Tampere, Finland
esa.rahtu@tuni.fi

Abstract—In this study, we investigate model inversion attacks against machine learning models in a black-box setting. On the one hand, an adversary can extract feature vectors of samples in a local dataset, while the underlying model's architecture and parameters are unknown. On the other hand, the adversary has illegitimate access to feature vectors of user data. We thoroughly analyze the following two attack scenarios on state-of-the-art models in person re-identification: recognizing auxiliary attributes and reconstructing user data. Extensive experiments show that the adversary could successfully infer sensitive information under severe constraints. Consequently, we highlight the importance of incorporating an encryption scheme when transferring and storing deep features. As an alternative to conventional encryption methods such as Advanced Encryption Standard (AES), we present a simple yet effective method termed ShuffleBits in which the binary sequence of each floating-point number gets shuffled. It serves as a plug-and-play module that is applicable to any neural network, and the model outputs encrypted data directly. The source code is available at https://github.com/nixingyang/ShuffleBits.

Index Terms—model inversion attack, black-box, encryption, neural network

I. Introduction

Due to the availability of large-scale datasets [1]–[3] and affordable computing resources, the field of machine learning has witnessed rapid progress over the past decade. Realworld applications can be found in everyday life, *e.g.*, targeted advertising in online shopping, recommender systems in video streaming services, and virtual assistants on smartphones. With the widespread adoption of techniques such as person reidentification [4]–[9], the concern over security issues can not be overemphasized. Since service providers process sensitive information from end-users, adversaries might misuse user data and compromise user privacy. Accordingly, significant efforts have been put into understanding the vulnerabilities in machine learning models [10]–[13]. In the following paragraphs, we outline four types of attacks that are predominant in the literature: adversarial example attacks, membership inference attacks, model extraction attacks, and model inversion attacks.

In adversarial example attacks, input data is slightly manipulated so that a human may not observe the changes while the model would make incorrect predictions [10]. In [14], a momentum term is integrated into the iterative process for performing attacks, and it stabilizes the direction for updates, avoids poor local maxima, and improves the success rate. Afterward, Su *et al*. [15] analyses an extreme case where only one pixel can be modified. Perturbation is encoded into an array, and the candidate solution is optimized by adopting differential evolution. By contrast, He *et al*. [16] generates an ensemble of weak defenses, and the resulting method does not always promote resilience to adversarial examples.

In membership inference attacks, an adversary is interested in identifying whether a specific sample is included in a model's training set [11]. Multiple shadow models are trained to simulate the target model while the membership in their training sets is available [11], [17]. Subsequently, a separate threat model is trained on the input-output pairs of the shadow models, and it behaves differently depending on whether the sample is used for training the target model. In [18], the relation between overfitting and membership vulnerability has been studied, and results indicate that overfitting is a sufficient but not necessary condition for membership vulnerability.

In model extraction attacks, an adversary has black-box access to a target model, and the primary objective is to duplicate the functionality of the target model [12]. Experiments on simple target models show that one could train substitute models locally on public datasets with near-perfect fidelity [12]. Under similar settings, a reinforcement learning approach is proposed in [19] to improve sample efficiency of queries, and a real-world image recognition model was pirated with reasonable performance. Juuti *et al*. [20] design a countermeasure that analyses the distribution of consecutive query requests and raises the alarm when suspicious activities are detected. Later on, two defense strategies are presented in [21]: the first membership inference strategy checks whether inputs are outliers, and the second watermarking strategy generates wrong outputs deliberately for a tiny fraction of queries.

In model inversion attacks, an adversary intends to infer input data from a released model [13]. Fredrikson *et al*. [22] managed to invert a linear regression model and predict the patient's genetic markers based on demographic information. With confidence scores returned by a facial recognition model, one could recover face images that are representative of a specific person in the training set [23]. In the case that only a partial prediction vector is returned, truncation is applied to feature vectors when training the inversion model in [24]. By contrast, Zhang *et al*. [25] shifts the focus to a whitebox setting and theoretically proves that the vulnerability to model inversion attacks is unavoidable for models with high predictive power.

Existing studies on model inversion attacks are subject to the following limitations: (1) The threat model is trained on the same dataset as the proprietary model [26]–[29]; (2) The adversary has white-box access to the proprietary model [25], [30]; (3) Experiments are limited to small-scale low-resolution datasets [23]–[25], [29]. To handle these problems, we investigate model inversion attacks in a more practical setting: (1) The proprietary dataset is unavailable, and the adversary has to collect and utilize a different local dataset; (2) Only a blackbox API is provided, while the architecture and parameters of the proprietary model are unknown; (3) Experiments are conducted on large-scale high-resolution datasets with state-ofthe-art proprietary models. In this study, our main contribution is twofold:

- We analyze two attack scenarios while avoiding the aforementioned limitations in previous works. Results show that it is feasible to recognize auxiliary attributes with decent accuracy and reconstruct user data that are recognizable. As a result, we give prominence to encrypting deep features.
- Different from conventional encryption methods, we propose an alternative scheme termed ShuffleBits. It can be implemented as a plug-and-play module inside neural networks, and only encrypted data leaves Graphics Processing Unit (GPU).

II. Proposed Method

A. Attack scenarios

Preliminaries. Figure 1 illustrates the background of attack scenarios in this study. A server runs a proprietary model which is trained on a proprietary dataset, and a response containing feature vectors is returned after processing a request containing user data. Additionally, the server's responses to the users are intercepted by the adversary, *i.e.*, feature vectors of user data are known to the adversary. Since the proprietary dataset is unreachable, the adversary collects and utilizes a local dataset instead. The purpose is to train a threat model that sniffs sensitive information of user data from the feature vectors.

Constraints. Multiple constraints complicate matters for the adversary. Firstly, the proprietary model and the threat model are trained on different datasets. Since samples from different datasets vary in terms of background, weather condition and camera angle, the domain gap would degrade performance. Secondly, the proprietary model's internal workings are out of reach because the adversary can only access it through a black-box API. Outputs of intermediate layers in the proprietary model are unattainable, and methods such as lateral shortcut connections [31] can not be applied. Thirdly, the threat model can not be optimized simultaneously with the proprietary model since the proprietary model is fixed. It leads to a mismatch between the objectives of the proprietary model and the threat model, *e.g.*, the proprietary model learns representative features for facial recognition while the threat model is trained to reconstruct face images.

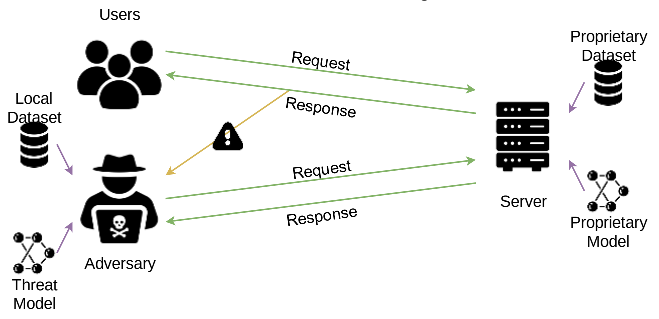


Fig. 1. Both the users and the adversary have access to the server, in which a proprietary model is deployed. Meanwhile, the adversary intercepts the server's responses to the users, and a local dataset is available.

Recognizing auxiliary attributes. Depending on the proprietary model in question, certain auxiliary attributes may be relevant. For example, one might be interested in a person's age and gender when using a facial recognition model. Although the original task (*i.e.*, recognizing faces) is inherently different from the auxiliary task (*i.e.*, predicting age and gender), the feature vectors for the original task may still contain relevant information for solving the auxiliary task. With a local dataset at hand, the adversary could annotate auxiliary attributes and construct a predictive model. The multilayer perceptron is suitable for solving such multi-label classification problems, where each sample is associated with multiple labels.

Reconstructing user data. One could interpret the whole system as an autoencoder. The proprietary model on the server is the encoder that maps raw data into feature vectors. The adversary builds a decoder that reconstructs raw data from feature vectors. The decoder is trained in an unsupervised manner, *i.e.*, it does not require a labeled dataset. The inputs are feature vectors extracted by the proprietary model, and the ground truth outputs are raw data. The decoder is optimized with an objective function so that the difference between ground truth data and reconstructed data is minimized.

B. ShuffleBits

In spite of recent studies on binarized neural networks [32], [33] which reduce memory consumption and improve inference speed, storing weights and activations in the singleprecision floating-point format is still the predominant option. Each single-precision floating-point value can be viewed as a 32-bit binary sequence (*i.e.*, binary32). The IEEE 754 standard [34] defines the procedure which converts a real number from decimal representation to binary32 format, and vice versa.

Given a single-precision floating-point value x, it can be represented as a finite binary sequence

$$(a_i)_{i \in I}, \tag{1}$$

where $a_i \in \{0,1\}$, $I = \{1,\ldots,n\}$ and $n = 32$.

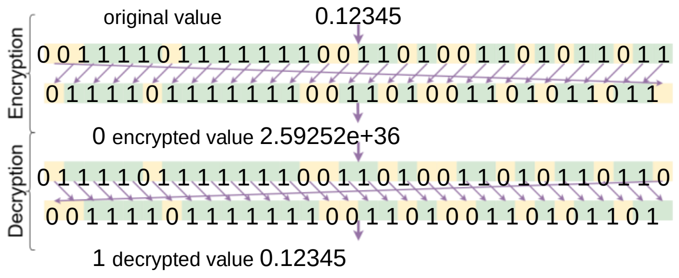


Fig. 2. A specific case of ShuffleBits: a left rotation operation is applied in the encryption process, and a right rotation operation is applied in the decryption process.

One could shuffle the original sequence according to an encryption key, and the encrypted sequence is

$$(b_j)_{j \in J}, \tag{2}$$

where $J = \{1,\ldots,n\}$. The encryption key is a bijective function f: $I \rightarrow J$, and it is an injective and surjective mapping of set I to set J. In addition, we have $b_{f(i)} = a_i$ for $i \in I$.

Similarly, the decrypted sequence is

$$(c_k)_{k \in K}, \tag{3}$$

where $K = \{1,\ldots,n\}$. The decryption key is another bijective function g: $J \rightarrow K$ which maps set J to set K. Furthermore, we have $c_{g(j)} = b_j$ for $j \in J$.

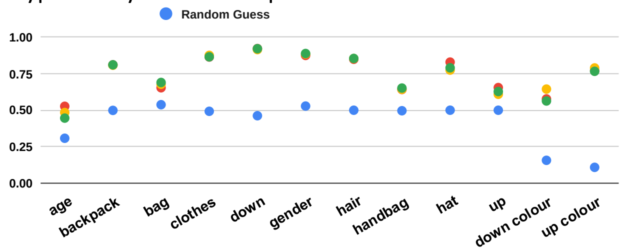
Since f is a bijection, it has an inverse function obtained by swapping the inputs and outputs in f. Let g be the inverse function of f, we have

$$a_i = b_{f(i)} = c_{g(f(i))} = c_i \text{ for } i \in I. \tag{4}$$

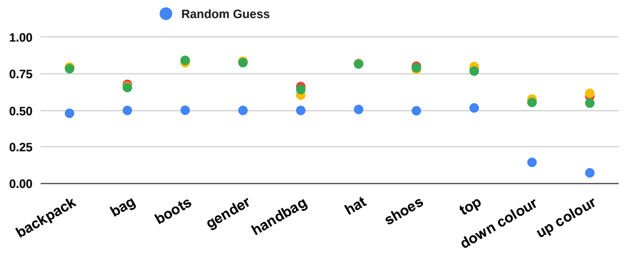
With the correct decryption key, it is apparent that the decrypted sequence is identical to the original sequence. Finally, the encrypted sequence and the decrypted sequence can be converted to decimal representation.

Figure 2 provides a step-by-step explanation of the proposed scheme using specific encryption and decryption keys. The original value's binary sequence is shuffled according to the encryption key, and the encrypted sequence corresponds to the encrypted value. By contrast, modifications are reverted so that the decrypted value is the same as the original value.

In the event of a brute-force attack, the adversary must systematically enumerate all possible decryption keys and check each of them. It is computationally infeasible to conduct exhaustive key search for the following reasons. Firstly, there are $32! \approx 2.63\text{e+}35$ unique keys. Thus the number of candidate keys is large. Secondly, it is not straightforward to validate whether the decrypted values are correct. Lastly, bit shuffling provides a one-time pad system in which the encryption keys in each request differ.



(a) Scores on the test set in Market-1501.



(b) Scores on the test set in DukeMTMC-reID.

Fig. 3. The balanced accuracies of each auxiliary attribute.

III. Experiments

A. Background

Domain. We conduct experiments in the domain of person re-identification, in which the objective is to retrieve a person of interest across multiple cameras [7].

Datasets. We select the following datasets that are widely used in recent works: Market-1501 [1], DukeMTMC-reID [2] and MSMT17 [3]. Each dataset has three partitions, namely, training set, query set, and gallery set. The latter two sets are merged as the test set. Throughout this study, we use MSMT17 as the proprietary dataset, while the local dataset is either Market-1501 or DukeMTMC-reID.

Models. The FastReID repository provides an unified instance re-identification library, along with a set of pre-trained models [8]. We include three top-performing methods which are built using the ResNet50 [35] backbone, namely, BoT [6], AGW [7] and SBS [8].

B. Recognizing auxiliary attributes

Model. For each auxiliary attribute, batch normalization [36] layers and fully connected layers are stacked to obtain the probabilities for each class. Similar to the proprietary models that classify person identities, we use only one batch normalization layer and one fully connected layer. Opting for this structure gives the best results in our experiments.

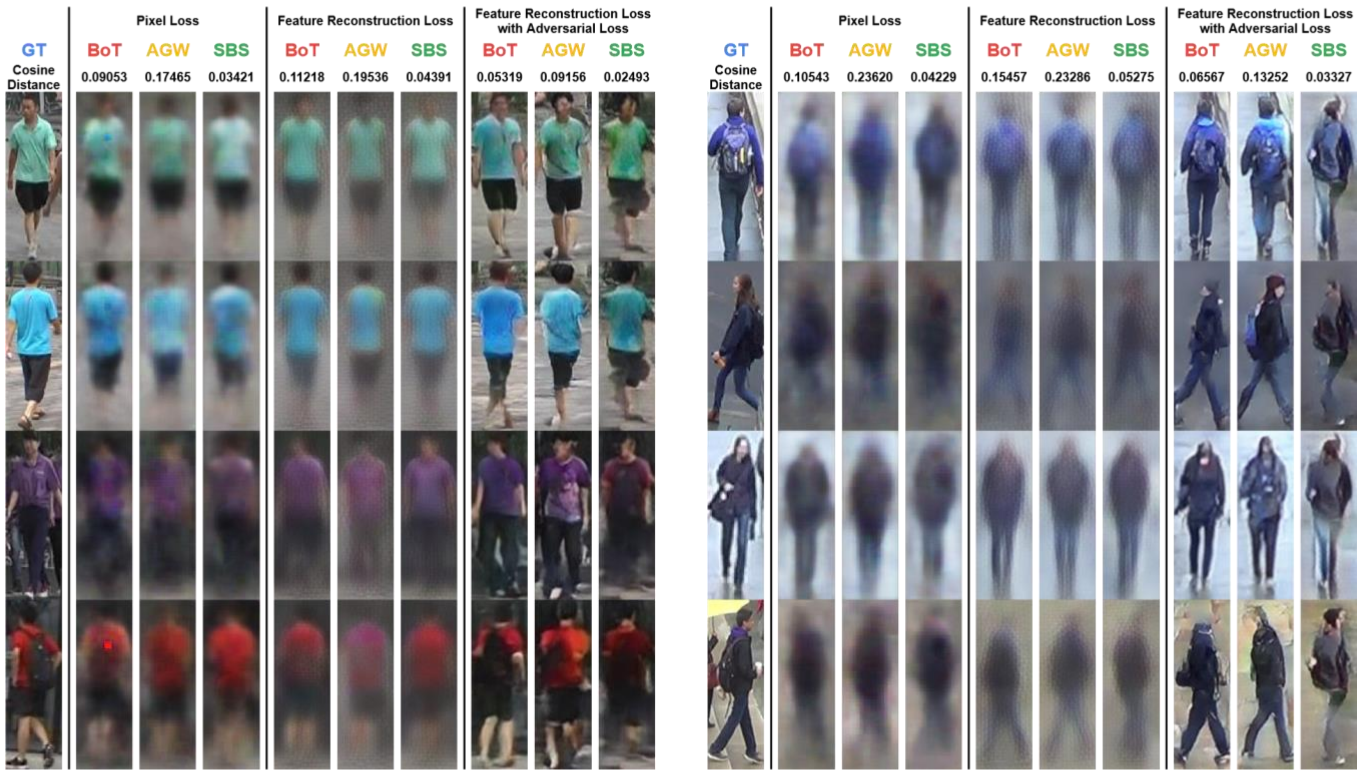
Loss function. The cross-entropy loss [37] is utilized to solve classification problems. Given an imbalanced dataset with unequal distribution of classes, classifiers would be biased in favor of the dominant classes. To address this issue, we assign a scalar value to each class during training so that more attention is paid to the under-represented classes [38]. The class weights are inversely proportional to the count number of occurrences of each class.

Evaluation metric. Conventional accuracy score measures the percentage of samples in which the predicted label matches the corresponding ground truth. However, it may give inflated performance estimates on imbalanced datasets. Thus, we adopt balanced accuracy [39] which is a better option, and it reports the average of recall calculated on each class.

Analysis. We leverage the auxiliary attributes in [5], and these annotations provide detailed local descriptions of pedestrians. Multiple labels are present while each label corresponds to a binary or multiclass classification problem. Figure 3 visualizes the balanced accuracies of each auxiliary attribute in two local datasets. Using feature vectors extracted by proprietary models yields significantly more accurate classifiers than guessing randomly.

C. Reconstructing user data

Model. Two sub-models are involved in reconstructing user data, and we utilize the generator and discriminator, which share similar architecture to the BigGAN [40] work. On the one hand, the generator maps feature vectors extracted by a proprietary model to images. Two fully connected layers followed by a reshaping operation generate the smallest feature maps, and five upsampling residual blocks increase the size to the target resolution. The last convolutional layer reduces the number of channels to 3, and the resulting predictions are in RGB color space. On the other hand, the discriminator classifies whether the images are original or synthetic. Five downsampling residual blocks with a global average pooling operation generate feature vectors of the images. Complemented with the feature vectors extracted by the proprietary model, a fully connected layer estimates the probabilities based on the concatenated feature vectors.



(a) Samples from the test set in Market-1501.

(b) Samples from the test set in DukeMTMC-reID.

Fig. 4. Comparison of the reconstructed images using different proprietary models, local datasets, and loss functions.

Loss function. The generator can be optimized with a weighted sum of the following loss functions: (1) The pixel loss [41] calculates the mean squared error between the ground truth images and the reconstructed images; (2) Given a pretrained model, one may extract an intermediate layer's outputs as feature maps. The feature reconstruction loss [41] refers to the mean squared error between the feature maps of the ground truth images and the reconstructed images. More specifically, we use the outputs of layer ”conv2 block3 out” in a ResNet50 [35] model which is pre-trained on ImageNet [42]; (3) The adversarial loss [43] measures how well the generator can fool the discriminator when feeding the outputs of the generator to the discriminator. By contrast, the discriminator is trained using the mean squared error loss proposed in [44]. Compared with the cross-entropy loss [37], it suppresses the vanishing gradients problem and stabilizes the learning process.

Evaluation metric. For each reconstructed image, the ground truth image is available. Instead of comparing these images pixel by pixel, we extract the reconstructed images' feature vectors using the same proprietary model and calculate the cosine distance between each feature vector pair. Note that the cosine distance metric is widely used when comparing two feature vectors in the inference procedure of person reidentification models. If the reconstruction is identical to the ground truth, it gives the minimum cosine distance 0. Additionally, comparing the cosine distance scores gives meaningful insights only if the proprietary model is the same, *i.e.*, one can not compare reconstructions generated from different feature embeddings.

Analysis. Due to the constraints listed in Section II-A, it is challenging to reconstruct user data precisely. Figure 4 illustrates the reconstructed images under various settings. Using the pixel loss gives blurry predictions. Switching to the feature reconstruction loss sharpens the images, while noticeable checkerboard artifacts can be observed. Nevertheless, such artifacts can be suppressed significantly by adding the adversarial loss, and the reconstructed images share strong similarities with the ground truth images. Additionally, combining the feature reconstruction loss and the adversarial loss results in the lowest cosine distance score between feature vectors of the ground truth images and the reconstructed images.

D. Importance of encrypting deep features

Results in Section III-B and III-C demonstrate that an adversary could successfully infer sensitive information under severe constraints. More specially, we manage to recognize auxiliary attributes with decent accuracy and reconstruct user data that are recognizable. In the presence of an encryption scheme, one has to include an encryption key in each request, and feature vectors in the corresponding response are encrypted (see Figure 1). Since the decryption key is being kept on the client side, the original values can be recovered without changes. The adversary could still train threat models on original feature vectors. However, the decryption key required to decrypt feature vectors of user data is unknown to the adversary, and threat models would not generate meaningful predictions on encrypted feature vectors. Note that the users and the adversary are using different encryption keys, and such encryption keys may vary in each request. Therefore, training threat models directly on encrypted feature vectors is not an option.

While AES [45] is widely accepted as the de facto standard for symmetric-key algorithms, the security of ShuffleBits is yet to be validated. The following three characteristics of ShuffleBits can be observed. Firstly, computations in ShuffleBits are inherently operations on tensors. As a result, it can be integrated into any neural network as a plug-and-play module without extra dependencies. Secondly, a model with ShuffleBits would output encrypted data directly, and only encrypted data leaves GPU. It reduces the risk of exposing unencrypted data. Thirdly, ShuffleBits can be seamlessly applied alongside conventional encryption methods. An adversary has to break all the encryption algorithms to get useful information, and such cascade encryption pipeline leads to better security.

IV. Conclusion

This study emphasizes the importance of encrypting deep features in the case of deploying machine learning models. Without a proper encryption scheme when transferring and storing deep features, an adversary could recognize auxiliary attributes and reconstruct user data, thus breaching user privacy. Additionally, we introduce the ShuffleBits method, which can be implemented as part of neural networks, and only encrypted data leaves GPU. A natural extension of this study would be to develop dedicated attacks against ShuffleBits from the perspective of cryptography.

References

[1] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-identification: A benchmark,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1116–1124.

[2] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking,” in *European Conference on Computer Vision*. Springer, 2016, pp. 17–35.

[3] L. Wei, S. Zhang, W. Gao, and Q. Tian, “Person transfer gan to bridge domain gap for person re-identification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 79–88.

[4] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation,” *arXiv preprint arXiv:1708.04896*, 2017.

[5] Y. Lin, L. Zheng, Z. Zheng, Y. Wu, Z. Hu, C. Yan, and Y. Yang, “Improving person re-identification by attribute and identity learning,” *Pattern Recognition*, 2019.

[6] H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang, “Bag of tricks and a strong baseline for deep person re-identification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, p. 0.

[7] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi, “Deep learning for person re-identification: A survey and outlook,” *arXiv preprint arXiv:2001.04193*, 2020.

[8] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, “Fastreid: A pytorch toolbox for general instance re-identification,” *arXiv preprint arXiv:2006.02631*, vol. 6, no. 7, p. 8, 2020.

[9] X. Ni, L. Fang, and H. Huttunen, “Adaptive L2 Regularization in Person Re-Identification,” in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 9601–9607.

[10] A. Kurakin, I. Goodfellow, S. Bengio, and others, “Adversarial examples in the physical world,” 2016.

[11] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[12] F. Tramer, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Stealing` machine learning models via prediction apis,” in *25th $\{$USENIX$\}$ Security Symposium ($\{$USENIX$\}$ Security 16)*, 2016, pp. 601– 618.

[13] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, “A methodology for formalizing model-inversion attacks,” in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 2016, pp. 355–370.

[14] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, “Boosting adversarial attacks with momentum,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[15] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[16] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, “Adversarial example defense: Ensembles of weak defenses are not strong,” in *11th $\{$USENIX$\}$ workshop on offensive technologies ($\{$WOOT$\}$ 17)*, 2017.

[17] Y. Long, V. Bindschaedler, and C. A. Gunter, “Towards measuring membership privacy,” *arXiv preprint arXiv:1712.09136*, 2017.

[18] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, “Privacy risk in machine learning: Analyzing the connection to overfitting,” in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 268–282.

[19] T. Orekondy, B. Schiele, and M. Fritz, “Knockoff nets: Stealing functionality of black-box models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4954–4963.

[20] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, “PRADA: protecting against DNN model stealing attacks,” in *2019 IEEE European Symposium on Security and Privacy (EuroS\&P)*. IEEE, 2019, pp. 512–527.

[21] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer, “Thieves on sesame street! model extraction of bert-based apis,” *arXiv preprint arXiv:1910.12366*, 2019.

[22] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in *23rd $\{$USENIX$\}$ Security Symposium ($\{$USENIX$\}$ Security 14)*, 2014, pp. 17–32.

[23] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[24] Z. Yang, E.-C. Chang, and Z. Liang, “Adversarial neural network inversion via auxiliary knowledge alignment,” *arXiv preprint arXiv:1902.08552*, 2019.

[25] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, “The secret revealer: Generative model-inversion attacks against deep neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 253–261.

[26] A. Dosovitskiy and T. Brox, “Generating images with perceptual similarity metrics based on deep networks,” *arXiv preprint arXiv:1602.02644*, 2016.

[27] ——, “Inverting visual representations with convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4829–4837.

[28] A. Mahendran and A. Vedaldi, “Visualizing deep convolutional neural networks using natural pre-images,” *International Journal of Computer Vision*, vol. 120, no. 3, pp. 233–255, 2016.

[29] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, “Conditional image generation with pixelcnn decoders,” *arXiv preprint arXiv:1606.05328*, 2016.

[30] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, “Dreaming to distill: Data-free knowledge transfer via deepinversion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8715–8724.

[31] H. Valpola, “From neural PCA to deep unsupervised learning,” in *Advances in independent component analysis and learning machines*. Elsevier, 2015, pp. 143–171.

[32] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[33] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1,” *arXiv preprint arXiv:1602.02830*, 2016.

[34] “IEEE Standard for Floating-Point Arithmetic,” *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[36] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[37] Z. Zhang and M. R. Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” *arXiv preprint arXiv:1805.07836*, 2018.

[38] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, “Multi-task learning and weighted cross-entropy for DNN-based keyword spotting.” in *Interspeech*, vol. 9, 2016, pp. 760–764.

[39] L. Mosley, “A balanced approach to the multi-class imbalance problem,” 2013.

[40] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.

[41] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[44] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794– 2802.

[45] J. Daemen and V. Rijmen, “AES proposal: Rijndael,” 1999.