Stealing Your Data from Compressed Machine Learning Models

Nuo Xu*, Qi Liu*, Tao Liu, Zihao Liu, Xiaochen Guo, Wujie Wen
Lehigh University
{nux219, qil219, tal519, zil719, xig515, wuw219}@lehigh.edu
*These authors contributed equally

***Abstract*—Machine learning models have been widely deployed in many real-world tasks. When a non-expert data holder wants to use a third-party machine learning service for model training, it is critical to preserve the confidentiality of the training data. In this paper, we for the first time explore the potential privacy leakage in a scenario that a malicious ML provider offers data holder customized training code including model compression which is essential in practical deployment. The provider is unable to access the training process hosted by the secured third party, but could inquire models when they are released in public. As a result, adversary can extract sensitive training data with high quality even from these deeply compressed models that are tailored for resource-limited devices. Our investigation shows that existing compressions like quantization, can serve as a defense against such an attack, by degrading the model accuracy and memorized data quality simultaneously. To overcome this defense, we take an initial attempt to design a simple but stealthy quantized correlation encoding attack flow from an adversary perspective. Three integrated components–data pre-processing, layer-wise data-weight correlation regularization, data-aware quantization, are developed accordingly. Extensive experimental results show that our framework can preserve the evasiveness and effectiveness of stealing data from compressed models.**

I. Introduction

Machine learning (ML), especially deep neural network (DNN), has nowadays become trustful and competent in applications such as image classification [1], speech recognition [2], and natural language processing [3]. However, training state-of-the-art DNN models requires not only expensive hardware platforms with substantial memory and computing resources, but also ML domain knowledge. To fulfill the ever-increasing need of "Plug and Play" DNN services on resource-constraint mobile, IoT and embedded devices for daily use, a common practice is to have models trained and optimized in cloud servers and then only execute the inference on local devices.

To develop customized ML applications, data holders can usually update their training dataset to the trusted third-party cloud servers, such as Google Cloud AI Platform [4], Amazon AWS [5], and Microsoft Azure ML Studio [6], and then select appropriate training algorithms, e.g. built-in-algorithms in the cloud or customized algorithms obtained from the open marketplace like Algorithmia [7], for model training. Hardware-oriented model compression techniques, such as quantization and pruning [8], for original model redundancy removal, can also be incorporated into these algorithms, to facilitate the fast and low-power inference on edge devices, where resource constraints are often enforced. Finally, the cloud provider will set up the training environment, assign computation resource, perform model training, and release different versions of trained models to fit various needs, e.g. quantized models with different levels of bit precision.

Despite the popularity of such a service model, it also raises confidentiality concern for data holders' training datasets which could contain clients' identity images, personal medical records, credit card numbers, etc., in privacy-sensitive applications. Many studies have revealed that the model itself can leak the private data during the training, such as model overfitting to unintentionally memorize massive information [9], model inversion attack to recover recognizable training images [10], membership inference attack to determine if the record belongs to the model's training dataset [11] etc. A recent study [12] further shows that a slight modification of the training algorithm (seemingly "normal") by a third-party ML algorithm provider can lead to stealthy and precise training data embedding into the model without harming the model performance, even the training environment and process are secured and isolated from such a malicious algorithm provider.

However, most of these studies are conducted based on the assumption that there always exists abundant model redundancy to memorize training information without accuracy loss. The redundancy, on the other hand, can be largely removed when applying hardware-oriented model compression techniques, which are essential to ease intensive computation and high memory overhead for inference on resource-limited platforms. As such, several interesting questions that naturally arise are: **1) Can existing compression techniques like quantization, help to prevent the training data leakage based on a trained model by removing the redundancy? If so, to what extent? 2) Is it possible to steal training data with high quality (effectiveness) from highly compressed models without accuracy loss (evasiveness)? If so, how will the adversary craft the seemingly normal training algorithm including compression to achieve this purpose?**

To answer these questions, in this paper, we **for the first time** investigate the potential privacy breach of training dataset when training compressed ML models that have limited or almost zero parameter redundancy. Specifically, we select a recent proposed strong attack which could well steal information by only correlating data with weight parameters under the guise of "regularization" during the training [12], and a representative quantization method [13], as a vehicle for this study. As we shall show in Table I, the quantization can serve as a defense mechanism to prevent such privacy leakage because of significantly degraded attack evasiveness (high model accuracy drop) and effectiveness (low quality of

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

embedded data) as the quantization bit width decreases. Based on this fact, we further explore whether there is any possibility to make such an attack still viable in deeply compressed ML models from an adversary perspective. Accordingly, three techniques which include fine-grained data pre-processing, layer-wise input-weight correlation regularization, and input-distribution aware weight quantization, can be easily developed and then seamlessly integrated into the original training pipeline to achieve both attack evasiveness and effectiveness simultaneously. Finally, our experiments on CIFAR-10 [14] and FaceScrub [15] datasets demonstrate the feasibility of training data stealing in deeply compressed ML models. We hope our study will enable the community to examine such an emerging privacy concern which could widely exist in training compressed models for many resource-limited devices.

II. Background and Motivation

A. DNN Model Compression

Model compression could squeeze out the redundancy of a DNN model by pruning unimportant connections (pruning) or reducing the bit precision of weight parameters (quantization) with marginal accuracy loss, so as to fit large models into resource-limited platforms. Particularly, weight quantization is usually a "must-have" step before deploying the model into any hardware [8], [13]. For example, deep compression [8] can linearly space the centroids in the range of original weights to initialize the shared weights and then quantize them into discrete values. The recent weighted entropy quantization [13] performs a non-linear quantization by considering each weight's contribution to final result and assigns more clusters for values that are neither too large nor too small. This process will involve light fine-tuning to boost accuracy. The method achieves adaptive quantization with flexible bit precision and can be easily deployed in pre-trained models. Without loss of generality, we select weighted entropy quantization as an example compression technique in this study.

B. Privacy Leakage

Machine learning model can exhibit incredible capability to memorize training data when data holders apply a malicious training algorithm which could stealthily encode the data into model parameters during the training with marginal accuracy loss [12]. The adversary can then extract such secret data from the parameters in the trained model once it is released.

LSB encoding attack directly replaces the least significant bits (LSBs) of trained model parameters with target bit string by leveraging model's inherent redundancy. Apparently, this attack can be easily defeated by the model compression (quantization) because of significantly increased accuracy sensitivity and decreased capacity for bit replacement.

Sign encoding attack relies on the sign bit of each parameter for bit embedding and is realized by adding a simple penalty term to the loss function, so as to force most sign bits of the parameters to follow the target secret bit string during the training. However, its attack efficiency is very low, as each parameter can only remember one bit.

Correlated value encoding attack is the strongest attack among three methods. This attack adds a malicious regularization term $C(\theta, s)$ in loss function to explicitly establish the correlation between training data s and model parameter θ:

$$C(\theta, s) = -\lambda_c \cdot \frac{|\sum_{i=1}^{\ell}(\theta_i - \overline{\theta})(s_i - \overline{s})|}{\sqrt{\sum_{i=1}^{\ell}(\theta_i - \overline{\theta})^2} \cdot \sqrt{\sum_{i=1}^{\ell}(s_i - \overline{s})^2}} \quad (1)$$

Here λ_c is the correlation rate to balance the encoded data quality and model accuracy. Enlarging λ_c can improve the former but decrease the latter. $\overline{\theta}$ and $\overline{s}$ are the mean of θ and s, respectively. ℓ represents the number of parameters.

For numerical data like image, this method can precisely encode the raw data into parameters by minimizing $C(\theta, s)$ (or rather the total training loss), and each pixel density can be decoded by simply remapping these parameters to values in the range of [0, 255]. Since the entire parameter can be used for data embedding during the training, this attack achieves the highest attack efficiency (more encoded data). Furthermore, it is possible that the established correlation can still survive after model compression.

C. Motivation

To explore whether quantization can mitigate the correlated value encoding attack, we apply weighted entropy quantization to ResNet-34 [16] trained with CIFAR-10 dataset under such an attack. Typical quantization bit widths (e.g. 8, 6, 4) are selected, in order to guarantee their corresponding benign models (without correlated value encoding attack) can maintain the similar accuracy acceptable by users, i.e., $\sim 90\%$. TABLE I reports the correlation attack model accuracy and the number of recognizable images by the model itself out of a total of 151 RGB images encoded in the model under three different correlation rates and quantization bits. For the same correlation rate, the attack model accuracy can be dramatically dropped as the quantization bit width decreases. This is because the redundancy carried by the correlation attack model is not sufficient when facing a very low quantization bit (e.g. 4), resulting in an unacceptable accuracy loss. Also, the number of recognizable images shows a similar trend because of encoded data quality drop caused by the distorted data-parameter correlation in a low quantization bit (visualized comparison can be observed from Fig. 5 using face images). The accuracy loss becomes more prominent for a same quantization bit width when the correlation rate becomes larger despite the increased number of recognizable images, e.g. 83.04%, 58 images at 4-bit and λ_c = 3 v.s. 75.46%, 75 images at 4-bit and λ_c = 10. Therefore, the quantized attack model neither maintains the accuracy (attack evasiveness) nor keeps the same amount of high-quality recognizable data (attack effectiveness). *This means existing model compression (e.g. weighted entropy quantization) can defeat such an attack at low bit precision.*

TABLE I: Model accuracy and recognizable image numbers of correlated value encoding attack after quantization.

Correlation Rate (λ_c)	**3.0**			**5.0**	**10.0**
Quantization Bit Width	8	6	4	4	4
Recognizable Images	88	82	58	59	75
Model Accuracy	88.79%	88.16%	83.04%	80.35%	75.46%

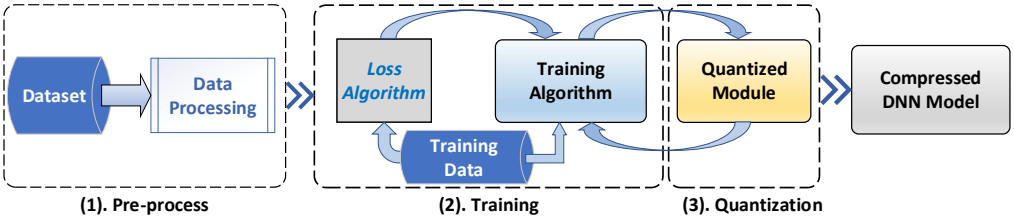

Fig. 1: Quantized correlation encoding attack flow

However, this does not mean that adversary is unable to conduct such an attack on compressed models if he or she slightly changes the quantization algorithm. *Our key observation is that if the quantization process can be guided by the statistical information (e.g. distribution) of target encoding data in a proper manner, then the quantized correlated model may not experience such significant parameter reshaping. As a result, both attack evasiveness and effectiveness can be guaranteed, making such an attack possible on highly compressed models.*

III. Overview

A. Threat Model

Our threat model is similar to, and extended from [12]. We assume that non-expert data holder will consume third-party training algorithms to generate customized ML models using his/her own dataset, which is sensitive and private. The data holder will conduct the training by using the provided malicious algorithm and confidential data in a trusted and secured platform with sufficient hardware resources like GPU clusters. Once the training is done, the data holder will validate the model with a test subset to measure the model accuracy, and only accept it if it passes the test with satisfying accuracy. Then data holder will publish the model for the user.

We assume the malicious ML training algorithm with the quantization process, is designed by the adversary. The training flow is the same as the benign routine except for very minor changes in the regularization and quantization that are necessary in normal training. The algorithm is executed on a secured third party platform that the adversary has no way to control, communicate with the training environment, or observe the training data during the whole training process.

The attack goal is to steal as much private training data as possible (e.g. embedding training images into the model with high data quality). In order to achieve this objective, the model needs to "memorize" the training data while passing the accuracy validation. Then the adversary can gain white-box access to the model after the data holder releases it, and examine all model parameters for information extraction. Note the adversary is unable to capture any temporary information or hyperparameter used during the training.

B. Attack Flow

Fig. 1 depicts an overview of our proposed quantized correlation encoding attack flow, which consists of three common steps often integrated in a training algorithm: data pre-processing, training with regularization, and quantization. Note the quantization will involve fine-tuning to compensate for the accuracy loss. The data pre-processing aims to guarantee the encoded data quality at the beginning by selecting a subset of the training dataset whose statistical information (e.g. pixel density distribution) can be similar to that of model parameter under correlation attack. It occurs automatically when the algorithm gets access to the training data. In the second step, a customized regularization term, of which different correlation rates λ are assigned to different layers based on their importance to accuracy and data encoding in correlation encoding attack, is proposed and included in the training loss. The last step is model quantization, which takes the statistical information of encoded training data into consideration during quantization to produces a compressed model that could preserve the secret data encoded in the training while maintaining the accuracy level required by the validation.

IV. Design

In this section, we present the proposed techniques in details following the attack flow.

A. Data Pre-processing

Since the basic idea of the correlated value encoding attack is to maximize the correlation between model weights and the target data, we expect that the distribution of trained weights under such an attack will be pushed towards a correlated distribution of the target dataset. Therefore, at the data pre-processing stage, the training algorithm should automatically select a subset of training data that follows a similar distribution for data embedding.

In this procedure, the algorithm will first cluster images based on the standard deviation (std) of the image pixel values as it can roughly represent the overall characteristic of an image. The mean of the standard deviation of the whole training dataset (std_{mean}) will also be calculated. Then a value range with length d around (std_{mean}) will be given: $std_{min} = \lfloor std_{mean} \rfloor, std_{max} = \lfloor std_{mean} \rfloor + d$. Image i that satisfies $std_{min} < std_i < std_{max}$ will be included in the candidate set S. The number of images that can be encoded (n) will be estimated based on the parameter amount and image size. Finally, the correlation target T will consist of n images randomly selected from the candidate set S.

Fig. 2(a) compares the weight distribution of the benign ResNet-34 model (without attack), and two malicious models with different correlation rates, and Fig. 2(b) shows the pixel value distributions of CIFAR-10 dataset with different std ranges. Based on Fig. 2(a), we observe that once the attack is launched, the distribution of the benign model (blue line in Fig. 2(a)) is significantly reshaped, and it is forced to be similar to the correlation targets' distribution (yellow line of Fig. 2(b)) as the correlation rate increases from 1 to 10. If we choose the candidate set range as [50, 55] ($std_{mean} = 50.36$), the selected

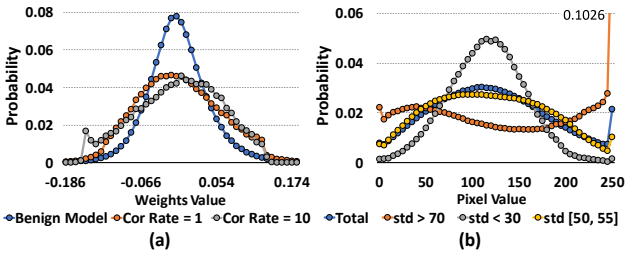

Fig. 2: (a) Parameter distributions of models with different correlation rates. (b) Pixel value distributions of images with different standard deviations.

target set exhibits a similar distribution as that of malicious models, e.g. yellow line of Fig. 2(b) v.s. orange (gray) line of Fig. 2(a). However, images with too large std (> 70) or too small std (< 30) have very different distributions.

B. Layer-wise Correlation Training Regularization

Once suitable encoding candidates from the training dataset are identified, the next step is to train the model with a fine-grained regularization term which assigns layer-wise correlation rate, to precisely embed these selected data into the model parameter with minimized accuracy loss.

Our observation is that each layer should have different contributions to final classification results, and the layers that are closer to the input, especially convolution layers for feature extraction, carry more importance than others in terms of accuracy. The weight distribution of each layer generally follows a Gaussian distribution, but each has a different weight value range for accuracy purpose. On the contrary, the correlated value encoding attack with a uniform correlation rate attempts to reshape the weights the same as the target data's distribution with the same value range. This leads to the conflict between model accuracy and encoding data quality. The problem becomes more aggravated after the model quantization. Therefore, we propose the following regularization term with customized correlation rates at different layer groups:

$$\mathrm{C}(\theta, s) = -\sum_{k=1}^{m}\left(\lambda_k \cdot \frac{|\sum_{i=1}^{\ell_k}(\theta_i-\overline{\theta}_k)(s_i-\overline{s}_k)|}{\sqrt{\sum_{i=1}^{\ell_k}(\theta_i-\overline{\theta}_k)^2}\cdot\sqrt{\sum_{i=1}^{\ell_k}(s_i-\overline{s}_k)^2}} \cdot P_k\right) \quad (2)$$

For each layer group $k \in m$, we have λ_k as the correlation rate, $P_k = \ell_k/\ell$ is the ratio of group k's number of weights ℓ_k to the total correlated weights amount ℓ . $\overline{\theta}_k$ and $\overline{s}_k$ are the mean value of the vector of secret values s and the weights θ at group k, respectively. In the extreme case, we can set $\lambda_k = 0$ for layers that are more sensitive to the modification of weights and not favorable for encoding. Consequently, both model accuracy and encoding quality can be improved by slightly reducing the total amount of encoded images.

We still use ResNet-34 and CIFAR-10 dataset for a case study. The layers are clustered based on the range of the weights. We choose the first 12 layers as group 1 and 13 to 16 layer as group 2, and the weights of these two groups encode the first 12% of images. The rest layers are categorized as group 3 to encode the remaining 88% of the images. TABLE II shows the percentage of badly encoded images distributed among the three groups with different correlation rates λ_c–3, 5 and 10. Here a badly encoded image is counted if the mean absolute pixel error (MAPE) between the decoded and original images is larger than 20. 100% images in group 1 and 75% images in group 2 are badly encoded at $\lambda_c = 3$. Increasing λ_c from 3 to 10 cannot improve the encoding efficiency for group 1, e.g. the percentage of bad images only drops from 100% to 48.1%, which is still much worse than that of group 3. This indicates that the weights of group 1 and 2 are naturally less correlated to the target data despite of a large correlation rate, and are difficult to change because of their importance to preserve the accuracy. In our final evaluation, we set the correlation rate of group 1 and 2 to 0 and use a different correlation rate for group 3, in order to achieve higher accuracy and better encoding quality in compressed models.

TABLE II: Number/Percentage of reconstructed images (bad, MAPE > 20) for models with different correlation rates

Correlation Rate (λ_c)	Total (453)	Group 1 (27)	Group2 (28)	Group 3 (398)
3.0	158 (34.9%)	27 (100%)	21 (75%)	110 (27.6%)
5.0	111 (24.5%)	20 (74.1%)	10 (35.7%)	81 (20.4%)
10.0	82 (18.1%)	13 (48.1%)	9 (32.1%)	60 (15.1%)

C. Target Correlated Quantization

The first two steps provide a well-trained attack model by establishing a strong correlation between the target data and model parameters. However, the weight quantization may still destroy the attack as discussed in Sec. II-C. Therefore, we further propose a quantization algorithm that can correlate the model parameters with target data's distribution.

Algorithm 1 shows the details of the proposed target correlation quantization. For a $\log_2 l$-bit precision (i.e., the quantization level is l) quantized model, we use correlated target information to decide the cluster boundary index of the weights. We first divide the pixel values of the correlated target image set into l clusters and count the histogram H (line3). Then H is used to determine the relative quantity of each cluster for the weights and the cluster boundary index b_0 to b_l (line 4 to 7). After obtaining the boundary index, the weights are sorted (line 8) and used to calculate the representative weight value r_i as well as the boundary weight value v_i for

Algorithm 1: Image-based Weight Quantization

1: **Input:** Correlation targets set T , Quantization level l, Total number of weights ℓ, weight list $[w_0 : w_{\ell-1}]$
2: **Output:** Quantized weight list $[q_0 : q_{\ell-1}]$
3: $H \leftarrow hist(T, l)$
4: $b_0 = 0$
5: **for** $i = 1$ *to* l **do**
6: $\quad b_i \leftarrow b_{i-1} + H[i-1] \times \ell$
7: **end for**
8: $S \leftarrow sort([w_0, ..., w_{\ell-1}])$
9: **for** $i = 0$ *to* $l-1$ **do**
10: $\quad r_i \leftarrow \frac{\sum_{j=b_i}^{b_{i+1}-1} S[j]}{b_{i+1}-b_i}$
11: $\quad v_i \leftarrow S[b_i]$
12: **end for**
13: $v_l \leftarrow \infty$
14: **for** $i = 0$ *to* $\ell - 1$ **do**
15: $\quad q_i \leftarrow f_q(w_i, [r_0 : r_{l-1}], [v_0 : v_l])$
16: **end for**

each cluster C_i (line 9 to 13), so that in each cluster C_i, weight value w satisfies $v_i \leq w < v_{i+1}$. Finally, we allocate all weights $[w_0 : w_{\ell-1}]$ to the corresponding cluster C_k based on the boundary value $[v_0 : v_l]$ and assign representative weight value r_k into the quantization list $[q_0 : q_{\ell-1}]$ (line 14 to 16).

Fig. 3 compares the weight distributions of the quantized attack model using the original weighted entropy quantization and our target-correlated quantization method. The original weighted entropy quantization significantly reshapes the weight distribution w.r.t. malicious models (Fig. 2(a)), thereby degrades the model accuracy to the degree that cannot be compensated by a retraining process. On the other hand, our method can well approximate the original distribution, which results in a well-maintained model accuracy and encoded data quality after the quantization.

V. Evaluation

A. Experiment Setup

Dataset and DNN Model: We adopt CIFAR-10 [14] and Facescrub [15] datasets in our experiment. CIFAR-10 consists of 60K 32x32 color images in 10 classes for image classification. For a comprehensive evaluation, its converted gray scale version is also included. Facescrub has 450 classes and more than 40K images for celebrity face recognition.

We use ResNet-34 [12] for CIFAR-10 image classification by following a similar configuration from [16]. The Inception-Resnet-v1 model [17] with modified softmax training algorithm is adopted in Facescrub recognition [18].

Methodology and Measurement: We integrate all our developed methods, including quantization, to establish the compressed correlation encoding attack flow. To create compressed models, we apply the proposed target correlated quantization and weighted entropy quantization with different bit precision.

We use the *mean absolute pixel error* (MAPE) to measure the quality of reconstructed image x' w.r.t. the original u-pixel image x: MAPE $= \frac{1}{u}\sum_{i=1}^{u}|x_i - x'_i|$. A lower MAPE value means better data quality. In addition, the *structural similarity index* (SSIM) [19] is used in our face recognition task to measure the reconstructed human face texture. Both *model accuracy* and *recognized image amount* are reported to evaluate the attack evasiveness and effectiveness.

B. Results

CIFAR-10 Classification. We target the images with *std* in [50, 55] and encode them into 17-34th layer (group 3) with three different correlation rates $\lambda_3 = 3, 5, 10$, and set $\lambda_1, \lambda_2 = 0$ for the 1-12th layer (group 1) and 13-16th layer (group 2) (see Sec. IV-B). TABLE III compares the model accuracy, MAPE, and recognized image amount (color image only) among the original attack model and our quantized attack model with different correlation rates and quantization bits. In most cases, our method can always maintain the accuracy of aggressively compressed models (even from 8 to 4 bits) at a similar level as that of uncompressed attack models for both gray and RGB versions. We also found that our method sometimes offers better accuracy, e.g. from 88.05% (original attack model with $\lambda_c = 5.0$) to 88.54% (8-bit quantized model with $\lambda_3 = 10$). This is because the improvement brought by data-preprocessing and layer-wise regularization can easily offset accuracy drop incurred by less aggressive quantization (e.g. 8-bit), leaving some margin to further enhance encoded data quality with a larger correlation rate.

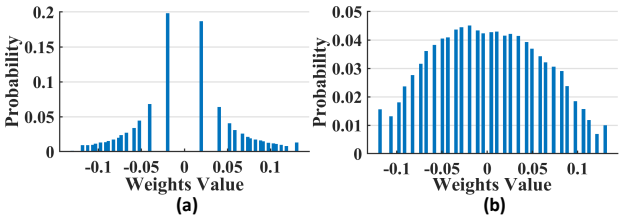

Fig. 3: Weight distributions of the quantized correlated value encoding attack model on 32 quantization levels: (a) Weighted Entropy Quantization (b) Target Image correlated Quantization

Although our method sacrifices some layers' capacity for data encoding due to introducing a zero correlation rate (e.g., 131 RGB images v.s. the original 151 RGB images), the encoding quality improvement guarantees that our encoded data can be as informative as that of the original attack models in almost all cases. For example, the recognizable image amount is similar to the uncompressed model, and sometimes even greater when the correlation rate is small, e.g. 111 vs. 98 with λ_3, λ_c = 3. Moreover, the significant MAPE reduction on reconstructed images can also be observed on our quantized DNN model for both gray and RGB images across all correlation rates. These results indicate that our method can not only secure the evasiveness of the attack with a guaranteed accuracy, but also improve the encoding efficiency with more informative data, while producing highly compressed models.

Fig. 4 compares the MAPE, accuracy and recognizable image amount among the original correlation attack model without quantization, with default 4-bit weighted entropy quantization, and our integrated attack flow with 4-bit quantization. We observe that the original correlation attack is not compression oriented. It suffers from significant accuracy degradation with weighted entropy quantization (red line with middle bar in all columns). The problem becomes even worse at a high correlation rate, as evidenced by the obvious accuracy

TABLE III: Results for original uncompressed attack models and our attack models under different configurations.

Model	**λ_c: 3.0**	**λ_1, λ_2=0, λ_3=3; std in [50, 55]**			**λ_c:5.0**	**λ_1, λ_2=0, λ_3=5; std in [50, 55]**			**λ_c: 10.0**	**λ_1, λ_2=0, λ_3=10; std in [50, 55]**		
Bit Width	Ori	8	6	4	Ori	8	6	4	Ori	8	6	4
MAPE (GRAY)	20.22	12.42	12.68	13.24	17.10	10.56	10.89	11.93	14.98	8.62	8.96	10.26
Accuracy (GRAY)	89.75%	89.72%	89.56%	88.31%	88.05%	88.73%	88.15%	87.81%	85.91%	88.54%	88.15%	87.31%
MAPE (RGB)	22.56	11.36	11.55	18.78	14.83	11.22	11.405	14.85	13.56	8.96	9.19	13.47
Accuracy (RGB)	89.82%	89.63%	89.52%	87.94%	88.16%	88.47%	88.19%	88.02%	86.69%	87.56%	87.50%	85.80%
Recognized Image Amount / Percent	98 (64.90%)	111 (84.73%)	111 (84.73%)	102 (77.86%)	112 (74.17%)	116 (88.55%)	115 (87.79%)	105 (80.15%)	127 (84.11%)	119 (90.84%)	115 (87.79%)	110 (83.97%)

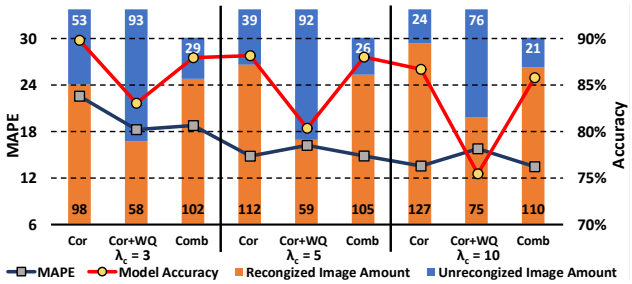

Fig. 4: The MAPE, accuracy and recognized image number of original correlation attack on uncompressed models (Cor), original correlation attack with weighted entropy quantization (Cor + WQ) and our method (Comb).

drop: from $\sim 90\%$ to $\sim 83\%$, $\sim 80\%$ and $\sim 75\%$ with λ_c = 3, 5, 10, respectively. In contrast, our method can better address this issue, by restoring the testing accuracy and significantly increasing the number of recognizable images (right orange bar v.s. middle orange bar in each group), even comparable with the uncompressed attack model (left orange bar in each group). These results show that the slight change of training pipeline with our method could make the training data stealing from highly compressed models possible.

Facescrub Recognition. Fig. 5 compares the reconstructed human face images with our quantized attack method and the original attack using default weighted entropy quantization. The visualized results clearly show that our method can well preserve the textures of the human face even on the 3-bit quantized model with only eight grayscale levels, significantly surpassing the original quantization (top row v.s. bottom row). Our method can reduce MAPE from 28.6 to 22.7 and increase the accuracy by 1.1% simultaneously, as TABLE IV shows. We also observe that more high-quality images (MAPE<20) can be reconstructed using our method. Besides, SSIM is used to better explain the visual texture difference in Fig. 5. As TABLE IV reports, with our method, more than $1/3$ of the reconstructed images (310 out of 924) can achieve SSIM>0.5, while there are only 12 such images with the original method.

VI. Conclusion

This work is the first to show it is feasible to steal high-quality information from securely trained compressed ML models for resource-limited devices. Our exploration shows that users' training data can be encoded into the model parameters with high quality, even on the extremely quantized DNN models, without compromising model accuracy. We propose

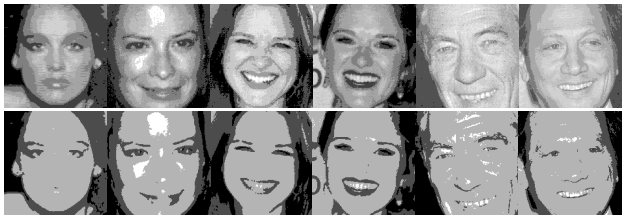

Fig. 5: Top row: Face images extracted from quantized model by our method; Bottom row: Face images extracted from quantized model by original weighted entropy method.

TABLE IV: Face recognition model with $\lambda_c = 10.0$ and quantization to 3-bits.

Model	Accuracy	MAPE	MAPE <20 Image Amount	Mean SSIM	SSIM >0.5 Image Amount
Uncompressed	95.30%	15.8	644	0.7088	718
Proposed Quantization	94.80%	22.7	468	0.4115	310
Original Quantization	93.70%	28.6	216	0.2976	12

the quantized correlation encoding attack flow and develop a set of quantization oriented techniques to demonstrate its feasibility through an end-to-end attack scenario. Our results show that the proposed method can maintain model accuracy, steal massive informative data from compressed models, and sometimes outperform the existing uncompressed attack routine. We hope our work can attract more follow up work to examine this emerging threat.

Acknowledgment

This work was partially supported by NSF Grants CNS-201126 and CCF-1750826, and 2019 FC^2 seed award.

References

[1] M. Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.
[2] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*, 2016, pp. 173–182.
[3] Y. LeCun *et al.*, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
[4] *Google Cloud AI Platform*, 2019, https://cloud.google.com/ai-platform/.
[5] *Amazon SageMaker on AWS*, 2019, https://aws.amazon.com/sagemaker/.
[6] *Microsoft AzureML Studio*, 2019, https://azure.microsoft.com/en-us/services/machine-learning-studio/.
[7] *Algorithmia*, 2019, https://algorithmia.com/.
[8] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
[9] C. Zhang *et al.*, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.
[10] M. Fredrikson *et al.*, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.
[11] R. Shokri *et al.*, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.
[12] C. Song *et al.*, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 587–601.
[13] E. Park *et al.*, "Weighted-entropy-based quantization for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5456–5464.
[14] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[15] H.W. Ng *et al.*, "A data-driven approach to cleaning large face datasets," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 343–347.
[16] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[17] D. Sandberg, *Classifier training of inception resnet v1*, 2018, https://github.com/davidsandberg/facenet/wiki/Classifier-training-of-inception-resnet-v1.
[18] F. Schroff *et al.*, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
[19] Z. Wang *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.