Privacy-Preserving Multimedia Mobile Cloud Computing Using Cost-Effective Protective Perturbation

Zhongze Tang
zhongze.tang@rutgers.edu
Rutgers University
Piscataway, NJ, USA

Zichen Zhu
zichen.zhu@rutgers.edu
Rutgers University
Piscataway, NJ, USA

Mengmei Ye
mye@ibm.com
IBM Research
Yorktown Heights, NY, USA

Yao Liu
yao.liu@rutgers.edu
Rutgers University
Piscataway, NJ, USA

Sheng Wei
sheng.wei@rutgers.edu
Rutgers University
Piscataway, NJ, USA

Abstract

Mobile cloud computing has been adopted in many multimedia applications, where resource-constrained mobile devices send multimedia data (e.g., images) to remote cloud servers to request computation intensive multimedia services (e.g., image recognition). Despite the performance improvement, the cloud-based mechanism often causes privacy concerns as the user data is offloaded to untrusted cloud servers. Existing solutions require computation-intensive perturbation generation on resource-constrained mobile devices. Also, the protected images are not compliant with standard image compression algorithms, leading to significant bandwidth consumption. We develop a novel privacy-preserving multimedia mobile cloud computing framework, namely PMC^2, to address the resource and bandwidth challenges. PMC^2 employs confidential computing on an edge server to deploy the perturbation generator, which addresses the on-device resource challenge. Also, we develop a neural compressor for the protected images to address the bandwidth challenge. Our evaluations of PMC^2 demonstrate superior latency, power efficiency, and bandwidth consumption while maintaining high accuracy in the target multimedia service.

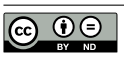
CCS Concepts

• **Security and privacy → Domain-specific security and privacy architectures**.

Keywords

Mobile cloud computing, protective perturbation

ACM Reference Format:
Zhongze Tang, Zichen Zhu, Mengmei Ye, Yao Liu, and Sheng Wei. 2025. Privacy-Preserving Multimedia Mobile Cloud Computing Using Cost-Effective Protective Perturbation. In *The 35th edition of the Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712678.3721879



NOSSDAV '25, Stellenbosch, South Africa

ACM ISBN 979-8-4007-1469-6/2025/03
https://doi.org/10.1145/3712678.3721879

1 Introduction

Mobile cloud computing has been adopted in many mobile multimedia applications to compensate for the limited computing resources on the mobile devices [17, 21, 29, 33]. In a typical mobile multimedia cloud computing system, the mobile device sends input data (e.g., images) to a remote cloud server, which provides computation-intensive services (e.g., image recognition) that are impractical on the mobile device due to resource limitations or intellectual property regulations. Mobile cloud computing has been adopted in many application domains, such as image recognition/annotation [29], object detection [21], and gaming [17], offering significant performance advantages.

However, mobile cloud computing can result in privacy concerns as the data offloaded (e.g., images) often involve privacy-sensitive components, which either the users themselves do not intend to release from their private possession [32] or disallow data sharing and dissemination due to privacy regulations [3, 15]. Particularly, visual privacy [23, 31, 32] is one of the most direct privacy issues faced by the end users when the data is offloaded to the cloud for processing. Visual privacy is concerned about the visual information contained in the disseminated data being visually perceivable by unauthorized individuals other than the data owner, even if there are no factual security or privacy breaches or side effects that can be attributed to the visual exposure. It has raised significant public concerns and discussions in the community [16, 20, 25, 27].

To address the visual privacy issue, protective perturbation-based approaches have been proposed recently, which inject noises to the private images to prevent the visual privacy exposure to human vision while preserving the capability of machine vision to complete the machine learning services [26, 28, 30, 31, 34]. However, such perturbation injection approaches often require executing computation-intensive models on the resource-constrained mobile device, posing significant challenges to latency, power consumption, and user experience. Therefore, it is challenging to deploy such privacy protection approaches on the mobile device in an end-to-end system.

We develop a privacy-preserving multimedia mobile cloud computing framework, namely PMC^2, to address the aforementioned challenges. PMC^2 adopts the state-of-the-art protective perturbation approach to protect the visual privacy of the private images but offloads the computation-intensive perturbation generation to a secure edge server to alleviate the computation workload on the

resource-constrained mobile device. The protected images generated by the secure edge server are then transferred to the remote cloud server for the desired multimedia service. To ensure the confidentiality and integrity of the perturbation generation process, PMC^2 employs hardware-based confidential computing techniques to deploy and execute the perturbation generation model in a secure container on the edge server. Moreover, to address the bandwidth challenge caused by the ineffective encoding of protected images, we develop a neural compressor that achieves a significantly higher compression ratio than standard image encoding methods while maintaining the target model accuracy.

We implement the proposed PMC^2 framework in an end-to-end multimedia mobile cloud computing system, involving a mobile device, a protective perturbation generator deployed on a secure edge server with confidential computing, and a regular cloud server running image recognition services. Our evaluations based on the CIFAR-10 dataset show superior latency, power, and bandwidth results of the proposed PMC^2 framework compared to the baseline system. Note that the perturbation generator and neural compressor required by PMC^2 can be trained without knowing the model structure/parameters or modifying the image recognition service model. Therefore, PMC^2 can be seamlessly deployed in empirical and legacy multimedia mobile cloud computing systems.

2 Background and Related Work

2.1 Multimedia Mobile Cloud Computing

A typical multimedia mobile cloud computing system involves a mobile client and a cloud server. The mobile client offloads input data to the cloud server to request computation-intensive services, which achieves significantly lower on-device resource (e.g., power) consumption and higher performance. However, the empirical deployment of such a mobile cloud computing system faces the following challenges:

- ***Privacy/Security challenge***. Since the privacy-sensitive user data must be offloaded to the cloud server, privacy concerns are often raised due to both the user's natural psychological needs and the legal regulations [4]. Also, the untrusted or unsecured cloud service provider and network may enable adversaries to obtain and abuse the private user data to issue various cybersecurity attacks [13].
- ***Bandwidth challenge***. The data offloading from the mobile client to the cloud server may consume a large amount of bandwidth in the communication network, especially when rich multimedia content is involved (e.g., high-definition images or videos).

2.2 Privacy-Preserving Protective Perturbation

Many research efforts have focused on privacy-preserving image delivery to address the visual privacy challenge, where the goal is to prevent the user's private image from being exposed to others. For example, several research works propose to inject perturbations to blur and protect the privacy-sensitive information in the images [26, 28, 30, 31, 34]. Such perturbation generator is typically deployed on the mobile client to protect the images to be offloaded to the cloud. Without loss of generality, we adopt the protective perturbation generator from [31] in our experiments, as it is fully open-source and does not require white box access to the proprietary image recognition model for training. By employing the proposed PMC^2 framework, our work aims to significantly reduce the power/latency overhead and the bandwidth consumption in these existing approaches.

Table 1: Results of using standard PNG and JPEG to compress regular and perturbed images.

Image Type	Encoding Methods			
	PNG		JPEG (no subsampling)	
	Size (Bytes)	Accuracy	Size (Bytes)	Accuracy
Regular	2662.60	94.24%	724.44	90.28%
Perturbed	2690.56	91.44%	1818.77	90.40%

2.3 Limitations of Existing Protective Perturbation and Image Encoding Approaches

We conduct two case studies to reveal the limitations of the existing protective perturbation and image encoding approaches.

2.3.1 Case Study 1: Excessive on-device power consumption and latency. We measure and compare the power consumption of the image recognition application [31] running on a Pixel 3 phone *with* and *without* injecting protective perturbation, as shown in Figure 1. The image recognition application involves a stream of 5,000 PNG images from the CIFAR-10 dataset [22], with a batch size of 256 for the perturbation generator. The target image recognition model is VGG13_bn. We observe that the average power consumption in the *with* perturbation case (the red line) is 4.961W, which is a 4.36x of 1.139W in the *without* perturbation case (the green line). Figure 1 also shows the latency results of the image recognition process, as indicated by the durations of the power traces. When the protective perturbation is added, the latency increases from 20.033s to 63.345s, which is a 3.16x slowdown. Furthermore, combining the power and latency results, the energy consumption of the privacy-preserving version of the application is around 13.78x of the original application without privacy protection. To summarize, the huge overhead of protective perturbation on mobile devices (i.e., 4.36x in power, 3.16x in latency, and 13.78x in energy) must be addressed to make the privacy protection approach practical.

2.3.2 Case Study 2: Ineffective compression for perturbed image. The standard image encoding mechanisms like PNG [2] and JPEG [1] are widely used in mobile cloud systems to efficiently transmit images. To identify the potential limitations in the case of the privacy-preserving images, we conduct a case study comparing the effectiveness of compressing regular and perturbed images using PNG and JPEG, with the target of achieving at least 90% of accuracy in image recognition, using VGG13_bn as the target image recognition model and 5,000 test images from the CIFAR-10 dataset. Table 1 shows the results of the case study. JPEG achieves 3.7x of compression ratio on regular images compared to lossless PNG. However, such compression ratio would decrease to 1.5x in the case of perturbed images in order to maintain a similar image

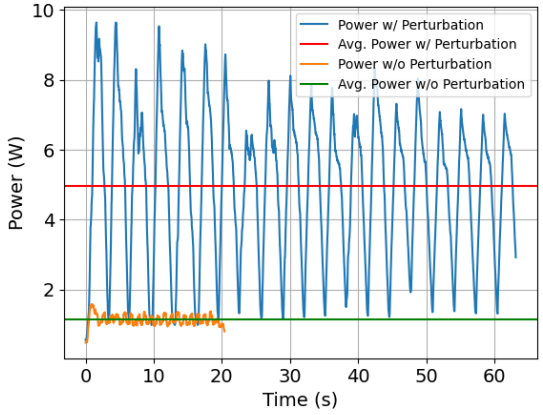


Figure 1: On-device power traces with & without privacy-preserving perturbation generation.

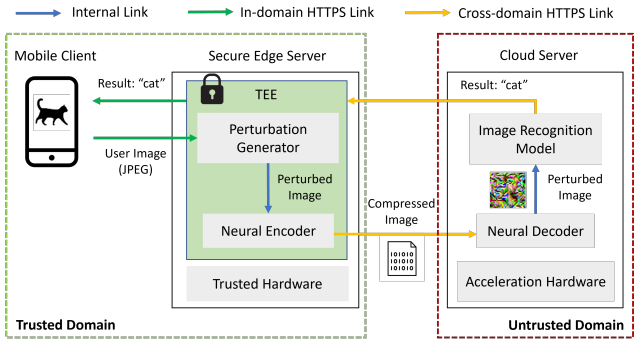


Figure 2: End-to-end workflow of the proposed *PMC*^{2} system.

recognition accuracy. The significant reduction in compression ratio indicates the ineffective compression achieved by the standard image encoding methods, which would cause higher bandwidth consumption in the delivery of the perturbed images.

3 *PMC*^{2} System Design

We develop a privacy-preserving multimedia mobile cloud computing framework, namely *PMC*^{2}, to proactively protect the privacy of user data in mobile-cloud image recognition, while consuming low on-device resources and network bandwidth for practical deployment. In a nutshell, the *PMC*^{2} system addresses the aforementioned ***privacy/security challenge***, as well as the associated power/latency overhead, by executing the security-sensitive and computation-intensive perturbation generation task on the trusted *secure edge server*, which is protected by the state-of-the-art confidential computing techniques and equipped with abundant resources for high performance computing. Also, it addresses the aforementioned ***bandwidth challenge*** by employing perturbation-aware neural compression for the perturbed images. Furthermore, the *deployment* of *PMC*^{2} system does not require modifications to the proprietary image recognition model, making it immediately deployable to empirical mobile cloud computing systems.

3.1 System Overview

Figure 2 shows the overall architecture of *PMC*^{2}. It introduces a trusted *secure edge server* between the *mobile client* and the traditional *cloud server*. The *secure edge server* accomplishes privacy-preserving perturbation generation in a confidential container. In addition, a neural encoder and a neural decoder are deployed on the *secure edge server* and the *cloud server*, respectively, to attain the bandwidth-saving goal.

The end-to-end *PMC*^{2} system works as follows to achieve privacy-preserving mobile-cloud image recognition. *First*, the *mobile client* compresses the privacy-sensitive images using JPEG and offloads them to the trusted *secure edge server* via an in-domain HTTPS link. *Second*, on the *secure edge server*, a perturbation generator runs inside the confidential container to securely inject protective perturbations to the received user images in real time; Then, the perturbed images are compressed by a perturbation-aware *neural encoder* and transferred to the *cloud server* for image recognition via the cross-domain HTTPS link. *Third*, the *cloud server* decodes the compressed images using a neural decoder. This process reconstructs the perturbed images in a manner similar to traditional image decoders like JPEG, which are agnostic to the user and do not leak any information about the protective perturbation. The reconstructed images are then fed into the machine learning-based image recognition model for computation-intensive classifications. *Finally*, the image recognition results are returned to the *mobile client* via the *secure edge server*.

3.2 Secure Protective Perturbation Generation

We employ the state-of-the-art confidential computing techniques to generate the protective perturbations on the *secure edge server* to address the ***privacy/security challenge*** without incurring power and latency overhead on the *mobile client*. Confidential computing has been adopted in the modern cloud platforms by leveraging trusted computing environment (TEE) technology [5–8]. Such technology provides secure enclaves in the system, which guarantees that the computation conducted in each enclave is isolated from the rest of the system.

To protect the confidentiality of the perturbation generation in *PMC*^{2}, we deploy the perturbation generator into a TEE on the *secure edge server*. Note that our proposed framework is generally compatible with any TEE techniques but, in this work, we leverage confidential containers (CoCo) [9] with AMD SEV [14]. In our implementation, we enable the AMD SEV functionality on the *secure edge server* to ensure the encryption and isolation for data in use. The *secure edge server* hosts the K8s control plane and confidential containers through the CoCo framework. Specifically, we untaint the *secure edge server* to make it available for pod scheduling. To execute the protective perturbation generator in CoCo, *kata-qemu-sev* runtime is installed and configured. Also, we use PyTorch 1.7.1 [24] inside CoCo for the perturbation generator.

3.3 Bandwidth-Saving Encoding

As shown in Figure 2, there are two communication links in *PMC*^{2} that consume network bandwidth: (a) in-domain, the compressed user image is transmitted from the *mobile client* to the *secure edge*

server, and (b) cross-domain, the encoded perturbed image is transmitted from the *secure edge server* to the *cloud server*. Let us denote the original image as Img_{orig}, which is encoded and stored in PNG by default. The JPEG-compressed version, $JPEG(Img_{orig})$, is delivered via the in-domain link. The perturbed image Img_{pert} is equivalent to $PertGen(JPEG(Img_{orig}))$, where $PertGen(\cdot)$ is the perturbation generator, and it will be encoded by the encoder function $Enc(\cdot)$. $Enc(Img_{pert})$ is the image transmitted via the cross-domain link.

Based on our experiments, the target model accuracy for $PertGen(JPEG(Img_{orig}))$ and $PertGen(Img_{orig})$ have very small differences (< 2%) with huge bandwidth savings (about 40%); therefore, the bandwidth-saving encoding for the in-domain link can be easily solved by adopting JPEG. To address the *bandwidth challenge* for the cross-domain link, i.e., $Enc(Img_{pert})$, we consider PNG and JPEG as the baseline $Enc(\cdot)$ approaches, which are subject to the compression inefficiency issue discussed in Section 2.3.2. PNG achieves lossless compression but with low compression ratio and high bandwidth consumption. For JPEG, we observe that the default *subsampling* parameters would break the functionality of the target model. When *subsampling* is disabled, to attain the same level of the target model accuracy, the compression ratio for perturbed images is significantly lower than that of regular images, as shown in Section 2.3.2. These observations motivate us to develop a new neural compressor tailored to the perturbed images to reduce the bandwidth consumption on the cross-domain link.

3.3.1 Baseline Approach: JPEG Compression. 4:2:0 chroma subsampling is used by default in libjpeg-turbo [11], a popular JPEG library. This subsampling makes chroma samples one-quarter of the luma ones. In our case study that uses JPEG to compress the protected image, we observe that the target model accuracy reduces significantly with chroma subsampling enabled. For example, when the quality factor is 89, the target model accuracy decreases from 91.17% to 11.50%, and the file size decreases from 2057.36 B to 1212.33 B. Although the file size is significantly reduced, the functionality of the target model is compromised. This indicates that the image recognition model recognizes the luma and chroma channels of the protected images equally and, when a lot of information is discarded in the chroma channel, the model cannot work properly. To maintain the high accuracy of the image recognition model and reduce the file size simultaneously, we disable the chroma subsampling (i.e., the sampling factor is 4:4:4) when using the JPEG encoding.

3.3.2 Proposed Approach: Perturbation-Aware Neural Compression. Figure 3 illustrates the workflow of the neural compression and decompression procedure in PMC^2, which is based on the Factorized Prior model [19]. In compression, the input image x is first encoded to latent y. The encoding step consists of three convolution layers, each followed by a Generalized Divisive Normalization (GDN) layer [18]. y is quantized to $\hat{y}$ to reduce the entropy (or the bit-rate). The EntropyBottleneck (EB) layer [19] utilizes the built-in probabilistic model to generate the compressed bit-string and the likelihood of $\hat{y}$ $L(\theta|\hat{y})$ under the parameters θ of the EB layer. In decompression, the compressed bit-string is decompressed by the EB first to recover $\hat{y}$, and the dequantization and the decoding steps convert it to the reconstructed image $\tilde{x}$. The decoding step is the inverse of the encoding, and it contains three deconvolution layers, each of which is followed by an inversed GDN layer.

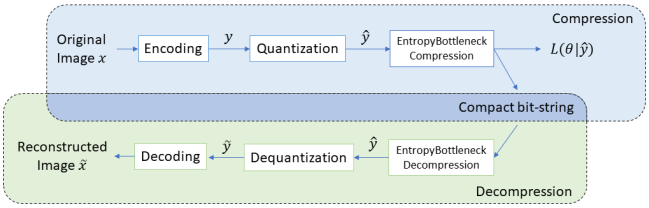


Figure 3: Workflow of the perturbation-aware neural compression and decompression.

In theory, the neural compressor used for PMC^2 (i.e., high compression ratio, low impact on the target model accuracy) should be trained by a loss function comprised of rate loss and accuracy loss. In this case, there is no need to optimize for distortion loss as it indicates the perceivable visual quality of a compressed image, which is not a concern in the context of compression for a machine learning service. The design of protective perturbation has resulted in a poor visual quality with most SSIM values below 0.02 [31], which is desired and unlikely to be restored by the neural compressor. In the experiments, we find that using the classical rate-distortion loss without any modification for training gives us a compressor that meets our rate-accuracy goal. In the rate-distortion loss, the distortion loss comes from the mean square error (MSE) between x and $\tilde{x}$, and we use bits per pixel (bpp) as the rate loss calculated by

$$rate\_loss = -\frac{\sum(\log_2 L(\theta|\hat{y}))}{num\_pixels} \tag{1}$$

where num_pixels is the total number of pixels in x.

4 Experimental Results

4.1 Experimental Setup and Implementation

***PMC*^{2} System and Baseline System Setup.** We deploy the end-to-end system as illustrated in Figure 2 to evaluate the performance of the proposed PMC^2 system. The *mobile client* is a Google Pixel 3 smartphone running Android 12, which has four 2.5GHz CPUs, four 1.6GHz CPUs, and a 2915 mAh battery. The *secure edge server* has an AMD EPYC 7443P Milan CPU with 24 cores and 48 threads, runs at 2.9GHz, and equips with 256 GB memory. Following the implementation of *secure edge server* described in Section 3.2, we adopt and deploy the open-source *perturbation generator* [31] to the confidential container of the *secure edge server*. We choose 4 target image recognition models and train their corresponding perturbation generators following the instructions provided in [31]. The auxiliary model used in the training and the target model accuracy of the test images are listed in Table 2. We run all the evaluations of perturbation generators using the batch size of 256. The target image recognition model runs on a *cloud server* with 8 vCPUs, 1 vGPU with one-third of the computational capacity of an NVIDIA A40 GPU, and 40 GB memory. The *mobile client* is deployed locally and connects to the Internet via an 802.11ac Wi-Fi, and the *secure edge server* and the *cloud server* are deployed

in different cloud centers [12]. Specifically, the *secure edge server* is located much closer to the *mobile client* than the *cloud server* to serve as an edge server. We adopt [31] as the baseline system, where the mobile client application and the server program run on the same mobile device and cloud server as in the PMC^2 system. All the data in both systems are transmitted over HTTPS.

Table 2: The target and auxiliary models used to train the perturbation generators and the target model accuracy.

Target Model	Auxiliary Model	Target Model Accuracy
VGG13_bn	ResNet18	91.52%
ResNet18	VGG13_bn	91.62%
DenseNet121	ResNet18	91.38%
GoogLeNet	ResNet18	91.22%

Dataset. We use the CIFAR-10 [22] dataset to train all the models and perform all the evaluations. The images are in the size of 32x32 and the average file size is 2662.56 Bytes. We use the 50,000 images in the training set of CIFAR-10 to train the 4 protective perturbation generators listed in Table 2, 5,000 images in the test set for validation, and the rest 5,000 images in the test set for evaluation. For each perturbation generator, we use the same images but with the particular perturbations added by the generator to train a dedicated Factorized Prior neural compression model [19].

4.2 Accuracy and Bandwidth Overhead

Table 3 presents the image recognition accuracy and the average size of the compressed image when deploying different image encoding methods between the *secure edge server* and the *cloud server*. The average size of $JPEG(Img_{orig})$ (i.e., the images sent from the *mobile client* to the *secure edge server*) and the average size of $Enc(Img_{pert})$ (i.e., the images sent from the *secure edge server* to the *cloud server*) are represented by S_1 and S_2, respectively. The standard deviation values of S_1 and S_2 are also provided after the ± symbols to describe the distribution of file sizes. All the standard deviation values are relatively small, indicating all the file sizes are generally close to the average values. The PNG-encoded Img_{pert} achieves recognition accuracies between 90.00% to 90.48%, and the image sizes range from 2544.70 to 2796.73 Bytes. The accuracy values decrease by 0.90% to 1.40% compared to the numbers reported in Table 2. We then take a step forward by replacing PNG with JPEG compression, which achieves significant data size reduction, ranging from 26.85% (ResNet18) to 32.37% (GoogLeNet), while maintaining similar recognition accuracies (i.e., above 90%).

The neural compressor in the proposed PMC^2 system shows superior accuracy and bandwidth savings. More specifically, the neural-compressed images still keep high target model accuracy, which is only reduced by 0.80% (GoogLeNet) to 1.42% (ResNet18) compared to the PNG encoding. On the other hand, a protected image can be compressed to a bit-string ranging from 8 to 8.29 Bytes, which is achieved by a neural compression model sized at only 12 MB. The data size is reduced by more than 99% compared to both PNG and JPEG encoding methods. In short, the proposed neural compression outperforms JPEG compression in terms of data size and thus bandwidth consumption between the *secure edge server* and the *cloud server*, without affecting the functionality of the image recognition model.

Note that the quality factors used for JPEG in both links, if applicable, are chosen experimentally to achieve similar target model accuracy (i.e., 90%) for a fair comparison. It is also worth noting that the average size of Img_{orig} is 2662.56 Bytes and, after applying JPEG compression with proper JPEG quality factors, the size (i.e., S_1) decreases by 40% on average, with almost no negative impact on the target model accuracy.

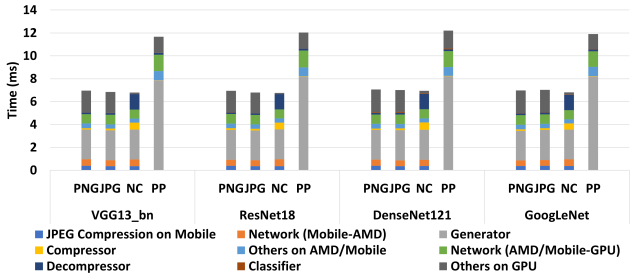


Figure 4: Timing performance of the PMC^2 system and the baseline protective perturbation system, employing 4 target models. Each target model is deployed in 4 different ways, including PNG, JPG (JPEG), NC (Neural Compressor), and PP (baseline protective perturbation system using PNG).

4.3 Timing Performance

We further evaluate the end-to-end latency of the proposed PMC^2 system using the 4 target image recognition models and image encoding methods. For comparison, we also conduct the same timing evaluation on the baseline protective perturbation system to uncover the gains of the proposed mechanism. Figure 4 shows the end-to-end latency results with timing breakdown analysis in 4 test cases for each target model. The first 3 test cases, namely *PNG*, *JPG*, and *NC*, represent using PNG, JPEG, and neural compression for the *secure edge server-cloud server* link in the PMC^2 system; the 4th test case, namely protective perturbation (*PP*), represents the baseline protective perturbation system with PNG encoding. The time breakdown results include 9 components, which are listed at the bottom of Figure 4. The *secure edge server* is named "AMD", and the *cloud server* is named "GPU". Particularly, there are no "JPEG Compression on Mobile"and "Network (Mobile-AMD)" components in the baseline system (i.e., *PP*).

The end-to-end latency of the PMC^2 system ranges from 6.766 to 7.070 *ms*, which is significantly less than the baseline protective perturbation system ranging from 11.668 *ms* to 12.212 *ms*. Among all 3 cases for PMC^2, the neural compression approach consistently outperforms the standard PNG by 1.80% (DenseNet121) to 6.17% (GoogLeNet), and it outperforms JPEG by 0.90% (ResNet18) to 6.42% (GoogLeNet). In general, PMC^2 takes 46.40% (VGG13_bn w/ neural compressor) to 48.67% (GoogLeNet w/ neural compressor) less time than the baseline system for the same target model.

Table 3: Accuracy and bandwidth comparison for different encoding methods across all target models. S_1 and S_2 represent the average file sizes over all 5,000 images from the *mobile client* to the *secure edge server* and from the *secure edge server* to the *cloud server*, respectively.

Target Model	Encoding Method								
	PNG			JPEG			Neural		
	S_1 (Bytes)	S_2 (Bytes)	Accuracy	S_1 (Bytes)	S_2 (Bytes)	Accuracy	S_1 (Bytes)	S_2 (Bytes)	Accuracy
VGG13_bn	1642.45±6.42	2691.38±6.79	90.30%	1854.95±6.69	1848.42±1.12	90.22%	1642.45±6.42	8.16±0.05	90.40%
ResNet18	1509.06±5.79	2544.70±3.27	90.22%	1509.06±5.79	1861.48±0.35	90.24%	1509.06±5.79	8.12±0.03	90.20%
DenseNet121	1572.23±6.17	2794.40±2.14	90.48%	1642.45±6.42	2042.20±0.65	90.38%	1572.23±6.17	8.29±0.07	90.42%
GoogLeNet	1572.23±6.17	2796.73±5.53	90.00%	1726.20±6.73	1891.50±1.06	90.32%	1726.20±6.73	8.00±0.01	90.42%

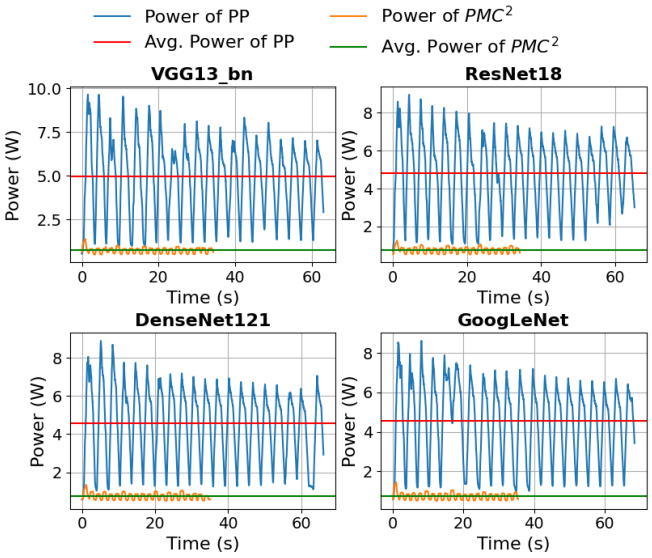


Figure 5: Power performance comparison between the baseline protective perturbation (PP) system and our *PMC*^{2} system using different target models. PNG is used in the baseline system, and neural compressor is used in *PMC*^{2}.

The "JPEG Compression on Mobile" takes about 0.37 *ms* on average, while the "Network (Mobile-AMD)" takes about 0.54 *ms* on average. On average, the protective perturbation generator takes 2.616 *ms* inside an SEV-enabled CoCo container or 8.149 *ms* on a smartphone to generate a protected image. The average PNG compression and decompression times are 0.125 *ms* and 0.145 *ms* in *PMC*^{2}, 0.269 *ms* and 0.164 *ms* in the baseline system, while those for JPEG are 0.149 *ms* and 0.156 *ms* in *PMC*^{2}. For both PNG and JPEG, compression and decompression take less than 4.45% of the total running time in *PMC*^{2}, and at most 3.49% in the baseline system. Other operations on the *secure edge server* take about 0.4 *ms* for both encoding methods, and other operations on the *cloud server* take about 1.8 to 3.1 *ms*. On the other hand, the compression and decompression times are 0.550 *ms* and 1.305 *ms* on average for the neural compressor. The neural compressor needs 0.362 *ms* for other operations on the *secure edge server* and 0.099 to 0.690 *ms* on the *cloud server*. It turns out that PNG and JPEG encoding/decoding take approximately equal time. The neural compressor needs about 4x time for compression and 9x time for decompression but less time for other operations to prepare the data for compression/decompression. The network latency and the image recognition model take stable and short time in all cases.

4.4 Power Consumption

We use the Monsoon power monitor [10] to collect the power traces of the mobile device, which runs in the same condition (e.g., network connection and screen brightness) for all test cases.

Figure 5 shows the power consumption comparison between the baseline protective perturbation system and the *PMC*^{2} system using 4 different target models. PNG is employed for the baseline system, and the neural compressor is adopted for the *PMC*^{2} system. The results for all the target models show a similar pattern that indicates significant power savings on the resource-constrained mobile device achieved by the proposed *PMC*^{2} system. For example, in the VGG13_bn case, the target image recognition model used in both systems is VGG13_bn. In the baseline system, most power traces fall into the 1.5 to 8.0 Watts range (blue lines), averaging 4.961 Watts (red lines). As a comparison, the *PMC*^{2} system consumes 0.751 Watts on average (green lines). Offloading the protective perturbation generation to the *secure edge server* makes the system much faster, as the test only lasts 33.956 seconds, almost half of the 63.345 seconds for the baseline system. Suppose we calculate the energy consumption by multiplying the average power and processing time; it turns out that the *PMC*^{2} costs only 8.11% energy of the baseline system. Similarly, the system can achieve 91.37% energy reduction when DenseNet121 is deployed as the target model.

5 Conclusion

We have developed a privacy-preserving multimedia mobile cloud computing framework, namely *PMC*^{2}, to address the privacy/security and bandwidth challenges in mobile image recognition. *PMC*^{2} offloads the perturbation generator to a secure edge server and employs a neural compression mechanism to effectively compress the perturbed images. Our evaluation results justify the superiority of *PMC*^{2} in terms of bandwidth, timing, and power consumption while maintaining high accuracy in image recognition. The repository of the project is at https://github.com/hwsel/PPMMCC.

Acknowledgments

This work was supported in part by the National Science Foundation under award 2350188.

References

[1] 1992. JPEG Standard. https://www.w3.org/Graphics/JPEG/itu-t81.pdf.

[2] 2003. Portable Network Graphics (PNG) Specification (Second Edition). https://www.w3.org/TR/2003/REC-PNG-20031110/.

[3] 2016. GDPR. Intersof Consulting. https://gdpr-info.eu.

[4] 2020. The Importance of Privacy - Both Psychological and Legal. https://www.psychologytoday.com/us/blog/emotional-nourishment/202007/the-importance-privacy-both-psychological-and-legal.

[5] 2021. Confidential computing: an AWS perspective. https://aws.amazon.com/blogs/security/confidential-computing-an-aws-perspective/.

[6] 2023. Azure confidential computing. https://azure.microsoft.com/en-us/solutions/confidential-compute/.

[7] 2023. Confidential Computing - Google Cloud. In *Google*. https://cloud.google.com/confidential-computing.

[8] 2023. Confidential computing with IBM. https://www.ibm.com/confidential-computing.

[9] 2023. Confidential Containers. https://github.com/confidential-containers.

[10] 2023. *High Voltage Power Monitor*. https://www.msoon.com/high-voltage-power-monitor.

[11] 2023. *libjpeg-turbo*. https://libjpeg-turbo.org/.

[12] 2023. Vultr. https://www.vultr.com.

[13] 2025. 40+ Alarming Cloud Security Statistics for 2025. https://www.strongdm.com/blog/cloud-security-statistics.

[14] 2025. AMD Secure Encrypted Virtualization. https://www.amd.com/en/developer/sev.html.

[15] 2025. Summary of the HIPAA Privacy Rule. https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html.

[16] Taslima Akter, Bryan Dosono, Tousif Ahmed, Apu Kapadia, and Bryan Semaan. 2020. "I am uncomfortable sharing what I can't see": Privacy concerns of the visually impaired with camera based assistive applications. In *USENIX Security Symposium (USENIX Security)*. 1929–1948.

[17] Ahmad Alhilal, Tristan Braud, Bo Han, and Pan Hui. 2022. Nebula: Reliable low-latency video transmission for mobile cloud gaming. In *ACM Web Conference (WWW)*. 3407–3417.

[18] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. 2015. Density modeling of images using a generalized normalization transformation. *arXiv* (2015).

[19] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. 2018. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436* (2018).

[20] Ankur Chattopadhayay and Isha Rijal. 2023. Towards inclusive privacy consenting for GDPR compliance in visual surveillance: a survey study. In *IEEE Annual Computing and Communication Workshop and Conference (CCWC)*. 1287–1293.

[21] Chien-Hung Chen, Che-Rung Lee, and Walter Chen-Hua Lu. 2017. Smart in-car camera system using mobile cloud computing framework for deep learning. *Vehicular Communications* 10 (2017), 84–90.

[22] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. (2009).

[23] José Ramón Padilla-López, Alexandros Andre Chaaraoui, and Francisco Flórez-Revuelta. 2015. Visual privacy protection methods: A survey. *Expert Systems with Applications* 42, 9 (2015), 4177–4195.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in neural information processing systems (NeurIPS)* 32 (2019), 8024–8035.

[25] Scott Pletcher. 2023. Visual privacy: Current and emerging regulations around unconsented video analytics in retail. *arXiv preprint arXiv:2302.12935* (2023).

[26] Zhongzheng Ren, Yong Jae Lee, and Michael S Ryoo. 2018. Learning to anonymize faces for privacy preserving action detection. In *European conference on computer vision (ECCV)*. 620–636.

[27] Abigale Stangl, Kristina Shiroma, Nathan Davis, Bo Xie, Kenneth R Fleischmann, Leah Findlater, and Danna Gurari. 2022. Privacy concerns for visual assistance technologies. *ACM Transactions on Accessible Computing (TACCESS)* 15, 2 (2022), 1–43.

[28] Qianru Sun, Liqian Ma, Seong Joon Oh, Luc Van Gool, Bernt Schiele, and Mario Fritz. 2018. Natural and effective obfuscation by head inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5050–5059.

[29] Yifan Tian, Yantian Hou, and Jiawei Yuan. 2017. CAPIA: Cloud assisted privacy-preserving image annotation. In *IEEE Conference on Communications and Network Security (CNS)*. 1–9.

[30] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. 2021. PECAM: privacy-enhanced video streaming and analytics via securely-reversible transformation. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*. 229–241.

[31] Mengmei Ye, Zhongze Tang, Huy Phan, Yi Xie, Bo Yuan, and Sheng Wei. 2022. Visual privacy protection in mobile image recognition using protective perturbation. In *ACM Multimedia Systems Conference (MMSys)*. 164–176.

[32] Ruoyu Zhao, Yushu Zhang, Tao Wang, Wenying Wen, Yong Xiang, and Xiaochun Cao. 2025. Visual content privacy protection: A survey. *Comput. Surveys* 57, 5 (2025), 1–36.

[33] Yi Zhao, Zheng Yang, Xiaowu He, Xinjun Cai, Xin Miao, and Qiang Ma. 2022. Trine: Cloud-edge-device cooperated real-time video analysis for household applications. *IEEE Transactions on Mobile Computing* 22, 8 (2022), 4973–4985.

[34] Bingquan Zhu, Hao Fang, Yanan Sui, and Luming Li. 2020. Deepfakes for medical video de-identification: Privacy protection and diagnostic information preservation. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*. 414–420.