

CNN-Based HVC Watermark Extraction for Print-and-Capture Scenarios

Fengshou Tao, Linghua Guo*, Xiaofeng Zhao

College of Bioresources Chemical and Material Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China

Abstract

Objective: This study addresses the difficulty of obtaining complete and clear digital extractions of halftone visual cryptography (HVC) watermarks under print-and-capture conditions. **Methods:** We constructed a differentiable print-and-capture degradation model that represented major physical degradation processes. Training samples were generated online by randomly combining degradation parameters. A convolutional neural network (CNNs) decoder was designed to recover 32×32 watermarks. The digitally stacked image was the primary input, while the registered watermarked image and digital key were auxiliary inputs. Multilevel downsampling established spatial correspondence between image features and watermark blocks. Multiscale dilated residual modules and channel attention extracted watermark features from degraded halftone dots. **Results:** The CNN decoder achieved a balanced correct decoding rate of 99.16% under simulated compound degradations. Its average decoding rate was 96.9% for real print-and-capture samples. This result exceeded direct digital stacking by 43.4 percentage points and blockwise XOR extraction by 13.8 percentage points. **Conclusion:** The CNN decoder learned the nonlinear mapping between degraded halftone-dot features and watermarks. It provides a practical approach for mobile HVC watermark extraction in print-and-capture scenarios.

Keywords

Halftone Visual Cryptography; Print-and-capture; Convolutional Neural Network; Printed-media Anti-counterfeiting.

1. Introduction

As digital printing and smart-device technologies continue to advance, printed products face increasing demands for copyright protection, identity authentication, and information traceability. These products include tickets, identity documents, commercial packaging, and anti-counterfeiting labels. Watermark extraction technologies based on smartphones and mobile platforms are now widely used for anti-counterfeiting verification, identity verification, and information traceability in printed products^[1].

Halftone Visual Cryptography (HVC) combines visual cryptography with halftoning. It enables computation-free visual decoding through the proper superposition of two images and has therefore been widely adopted^[2]. However, mobile extraction in practice is affected by variations in shooting distance and angle, nonparallel paper and imaging planes, uneven illumination, lens blur, image resampling, and compression. These factors jointly introduce geometric and photometric distortions. For watermarks carried by halftone dot structures, such distortions can also disrupt screen-phase relationships and local dot arrangements. As a result, watermark contours become blurred and fine details are lost. Achieving complete and clear digital extraction of HVC watermarks under print-and-capture conditions is therefore a key challenge for mobile applications.

The key to HVC watermark extraction is to establish spatial correspondence between a watermarked halftone image and a share image, reference key, or decoding template[3]. The method then converts changes in local dot density, dot position, texture orientation, and screen frequency or phase into observable or computable watermark responses. Existing extraction methods can be broadly divided into three categories.

The first category comprises optical-superposition and logical-differencing methods. Fu and Au proposed a conjugate error-diffusion data-hiding method that embeds binary patterns into two error-diffusion halftone images. Hidden patterns are recovered through precise image superposition^[4]. Guo, Au, and Tang proposed CoP-MED, which embeds secret patterns into two multiscale error-diffusion halftone images. Extraction is performed through image superposition or pixel-wise XNOR operations^[5]. Wang et al. jointly optimized two DBS halftone images using ADBS and DADBS. This approach keeps the watermark hidden in each individual image and reveals it after superposition^[6].

The second category comprises spatial-correlation and template-matching methods. Rosen and Javidi encoded hidden information in the positions of dots within halftone cells. They used a spatial correlator to detect the correlation response between the dots and a reference filter^[7]. Basak and Gaurav used periodic line gratings matched to the screening angle. They extracted the watermark from luminance and texture differences produced after superposition^[8]. Guo and Seshathiri combined correlation analysis with halftone statistical features to detect local dot structures and decode binary watermarks^[9].

The third category comprises screen-ruling and phase-demodulation methods. Oztan and Sharma used continuous phase modulation to encode information in screen phase or spatial frequency. They recovered the watermark from the reference screen ruling and phase response^[10]. Sharma combined screen periodicity with color-spectral differences to estimate CMYK halftone separations from RGB-scanned images. The watermark was then detected in each color separation^[11]. Ulichney et al. used spectral features to estimate the position, scale, and orientation of patterns captured by mobile devices. These estimates provide parameters for subsequent geometric correction and information extraction^[12].

Together, these methods represent the main technical pathways for halftone watermark extraction. However, even when existing registration methods restore the geometric correspondence between images after transmission through a print-and-capture channel, dot structures degraded by blur, resampling, and photometric changes remain difficult to recover. Watermark visibility and extraction quality therefore remain compromised.

In recent years, deep learning has been applied to digital watermark embedding and extraction under physical-channel conditions. StegaStamp improves watermark adaptability to printing and photographing by simulating perspective transforms, color variations, blur, and noise during training^[13]. Fang et al. developed a screen-camera robust watermarking method by constructing a screen-display-to-camera acquisition degradation model. The model learns the mapping between degraded images and watermark information^[14]. Qin et al. established a deep noise-simulation model for the print-and-capture process. It reduces the discrepancy between digitally generated training data and images acquired under real conditions^[15]. RoPaSS simulates multiple degradations in real physical channels to improve watermark recovery under complex acquisition conditions^[16].

These studies indicate that convolutional neural networks (CNN) can learn nonlinear relationships between compound-degraded images and hidden information. Nevertheless, existing deep watermarking methods mainly target continuous-tone images or general physical watermarks. Research on recovering watermarks from degraded halftone dot structures remains limited.

Based on this analysis, we propose a CNN-based method for extracting HVC watermarks in print-and-capture scenarios. Mobile-acquired images are first preprocessed, after which a CNN decoder performs digital extraction of degraded HVC watermarks. The results show that the proposed method improves the completeness and clarity of HVC watermark extraction. It provides a new technical approach for mobile applications of halftone visual cryptography in print-and-capture settings.

2. CNN-Based HVC Watermark Extraction for Print-and-Capture Scenarios

2.1 Overall Framework

We developed an integrated method for digital HVC watermark extraction in print-and-capture scenarios. The method comprises image preprocessing, training-data construction, and CNN-based watermark decoding. First, the watermarked region is located in the smartphone image, followed by perspective correction and fine registration. These operations establish spatial correspondence between the captured image and the digital key. Second, a print-and-capture channel degradation model generates online training samples containing multiple physical distortions. Finally, the digitally stacked image, registered image, and digital key are jointly input into the CNN decoder to recover a 32×32 block-level watermark. Figure 1 shows the overall workflow.

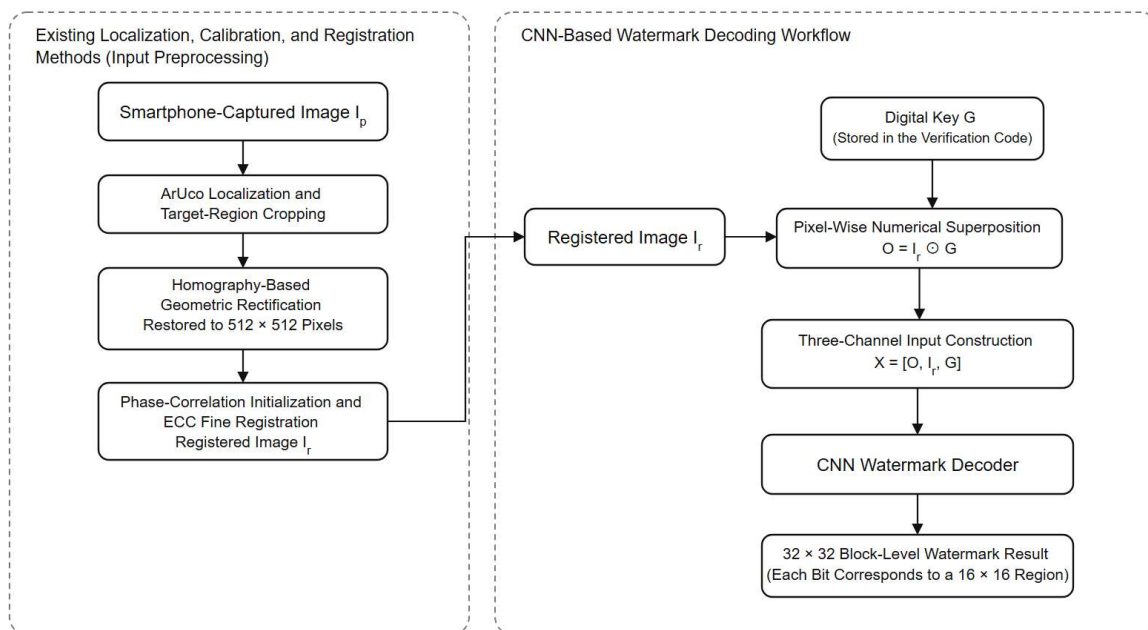


Figure 1. Overall framework of the HVC watermark extraction method

HVC watermarking generally encodes a binary watermark into two matched halftone shares. The watermark is difficult to recognize from either share alone. Correct spatial superposition changes the local dot-coverage relationship and reveals the hidden watermark. Instead of producing a physical transparent share, we register and digitally stack the printed-and-captured watermarked halftone image with its digital key. This procedure enables automated digital decoding of the HVC watermark. The method handles geometric correction and dot-degradation compensation separately. Image preprocessing reduces geometric deviations introduced during capture. The degradation model generates training samples covering different printing and capture conditions. The CNN decoder learns the nonlinear mapping between damaged halftone dots and the block-level watermark.

2.2 Image Preprocessing

HVC embedding encodes a binary watermark into two matched halftone shares by adjusting local dot arrangements and coverage relationships. Consequently, the watermark remains difficult to observe in either share alone. Correct spatial superposition changes the local dot density or pixelwise logical relationship and reveals the hidden watermark. In this study, the printed-and-captured watermarked halftone image is stacked with the corresponding digital key defined by the HVC embedding rule.

Image preprocessing removes irrelevant backgrounds, corrects major capture-induced geometric distortions, and produces a normalized image aligned with the digital key. The procedure sequentially

includes watermarked-region localization and cropping, coarse homography correction, and fine registration using phase correlation and enhanced correlation coefficient (ECC) maximization.

Let the raw smartphone image be I_p , and let the digital key matching the embedding rule be G . The watermark region is first located and cropped from I_p to remove irrelevant background. Existing geometric-correction methods then correct translation, rotation, scale variation, and perspective distortion, producing the normalized image I_c . An existing image-registration method subsequently estimates the relative spatial transformation between I_c and the digital key G . The captured image is transformed into the key's reference coordinate system to obtain the registered image I_r , expressed as:

$$I_r = \mathcal{R}(I_c, G) \quad (1)$$

Where $\mathcal{R}(\cdot)$ denotes the existing image-registration operation. Registration reduces spatial deviations between the captured image and the digital key. It therefore maintains correspondence between their dot positions and screen structures. This study does not propose new localization, geometric-correction, or registration algorithms. These steps serve only as input preprocessing for CNN-based watermark decoding.

2.3 Construction of the Training Dataset

2.3.1 Print-and-Capture Channel Degradation Model

Let the ideal digitally watermarked HVC halftone image be $I^{(0)}$. We modeled the print-and-capture process as seven consecutive differentiable degradation operations. This design reduces the distribution gap between digital training samples and real smartphone images. The degradation parameters and activation probabilities were sampled independently for each sample. The resulting combinations simulated distortions caused by different printers, capture poses, illumination conditions, and image-encoding settings. The print-and-capture process is expressed as:

$$I^{(7)} = \mathcal{J}_q \circ \mathcal{N}_{\sigma_n} \circ \mathcal{L}_{a,c,\gamma} \circ \mathcal{S}_s \circ \mathcal{H}_{\theta} \circ \mathcal{B}_{\sigma_b} \circ \mathcal{D}_{\rho} (I^{(0)}) \quad (2)$$

Here, $\mathcal{D}_{\rho}$, $\mathcal{B}_{\sigma_b}$, $\mathcal{H}_{\theta}$, $\mathcal{S}_s$, $\mathcal{L}_{a,c,\gamma}$, $\mathcal{N}_{\sigma_n}$ and $\mathcal{J}_q$ denote ink spread, lens blur, homographic distortion, image resampling, illumination and contrast variation, sensor noise, and JPEG compression, respectively. The corresponding degradation parameters are ρ , σ_b , θ , s , a , c , γ , σ_n , and q . The parameters were sampled independently for each sample to generate compound degradations under varied printing and capture conditions.

The individual degradation processes are described below.

(1) Ink Spread and Dot Gain

During printing, toner or ink spreads into adjacent regions, making actual black dots larger than their ideal digital counterparts. We applied morphological dilation to the black-dot mask to simulate dot-area expansion. The degradation strength was controlled by ρ , with an upper limit of 0.5. This operation mainly altered dot areas and edge shapes, representing a major printing-induced change in halftone structure.

(2) Lens and Printing-System Blur

Print-system dot spread and smartphone-lens defocus transform ideal binary dot boundaries into continuous grayscale transitions. We simulated this process using Gaussian filtering. The Gaussian-kernel standard deviation σ_b was randomly sampled from $[0.4, 1.2]$:

$$I^{(2)} = \mathcal{B}_{\sigma_b} \left(I^{(1)} \right) \quad (3)$$

(3) Homographic Transformation

During smartphone capture, the paper and imaging planes are rarely parallel. Changes in distance and angle therefore cause translation, rotation, scaling, and perspective distortion. We combined these geometric changes into one homography. The transformation included translations up to 8 px, random rotation, scale changes within $\pm 2\%$, and corner perturbations up to 4 px:

$$I^{(3)} = \mathcal{H}_{\theta} \left(I^{(2)} \right) \quad (4)$$

Combining all geometric changes into one transformation reduces repeated damage to halftone-dot structures from multiple interpolation operations.

(4) Image Resampling

Sensor sampling, capture-distance variation, and image resizing alter the effective sampling density of halftone dots. Images were randomly downsampled to a minimum scale of 0.70 and then restored to 512×512 pixels. This operation simulated resolution loss and interpolation errors during image acquisition:

$$I^{(4)} = \mathcal{S}_s \left(I^{(3)} \right) \quad s \geq 0.70 \quad (5)$$

(5) Illumination and Contrast Variation

Captured images are affected by ambient illumination, paper reflectance, toner color, smartphone auto-exposure, and image-processing algorithms. We constructed a low-frequency illumination field with a maximum amplitude of 0.2. Image contrast and gamma were also randomly adjusted. These operations simulated uneven illumination, nonuniform paper reflection, imperfectly black toner, and variations in smartphone brightness response:

$$I^{(5)} = \mathcal{L}_{a,c,\gamma} \left(I^{(4)} \right) \quad (6)$$

(6) Sensor Noise

Smartphone cameras and image-signal processing introduce random noise. We added Gaussian noise with a standard deviation not exceeding 0.05:

$$I^{(6)} = I^{(5)} + n, \quad n \sim \mathcal{N} \left(0, \sigma_n^2 \right) \quad (7)$$

(7) JPEG Compression

Mobile images commonly undergo JPEG encoding or platform compression, which alters dot edges and local high-frequency textures. We applied JPEG compression with the quality factor q randomly sampled from a predefined range. A straight-through estimator approximated the gradient of JPEG quantization, allowing this compression operation to be used during network training:

$$I^{(7)} = \mathcal{J}_q(I^{(6)}) \quad (8)$$

The seven stages represent printing, optical imaging, capture pose, image sampling, photometric response, sensor acquisition, and image encoding. The first four stages mainly modify dot positions and edge shapes. The remaining stages mainly alter grayscale values, local contrast, and high-frequency details. Random combinations broaden the effective training distribution and help the CNN learn watermark representations robust to compound print-and-capture distortions.

2.3.2 Sample Generation and Dataset Partitioning

Training samples for the CNN decoder combined ideal carrier images, randomly generated HVC watermarks, and the print-and-capture channel simulator. We used 5,000 natural images from COCO val2017. At the original-image level, 4,000 images were assigned to training, 500 to validation, and 500 to fixed testing. This partition prevented degraded versions of the same carrier image from appearing in both training and test sets.

For each carrier image, the adopted HVC embedding method first generated an ideal watermarked halftone image $I^{(0)}$ and its corresponding digital key G . Training watermarks were generated randomly to vary foreground position, shape, and local structure. The validation and test sets additionally contained real logo patterns unseen during training, allowing assessment of generalization to novel watermark shapes. The seven-stage degradation model then transformed $I^{(0)}$ into a simulated smartphone image $I^{(7)}$.

To match the actual decoding workflow, the degraded image was normalized as described in Section 2.2 and registered with the digital key before entering the CNN. The registered image is denoted by I_r . The digitally stacked image was then calculated according to the HVC stacking rule:

$$O = I_r \odot G \quad (9)$$

The images O , I_r , and G were concatenated along the channel dimension to form the CNN input:

$$X = \text{Concat}(O, I_r, G) \in \mathbb{R}^{512 \times 512 \times 3} \quad (10)$$

The ideal 32×32 watermark corresponding to the input image served as the supervisory label and is denoted by W :

$$(X, W) = (\text{Concat}(O, I_r, G), W) \quad (11)$$

The watermark measured 32×32 pixels, whereas the input image measured 512×512 pixels. Each watermark bit therefore corresponded to a 16×16 -pixel region in the carrier image. The training objective was to recover the original 32×32 watermark from the three-channel input under print-and-capture degradations.

Training samples were generated online. At each iteration, a carrier image and watermark pattern were randomly selected, and the seven-stage degradation parameters were resampled. The same carrier image could therefore produce different watermark content and channel distortions across iterations. Compared with fixed pre-generated samples, online generation broadened the effective distribution and reduced memorization of specific degradation patterns.

We retained 10% degradation-free samples to prevent the network from learning only watermark features under compound degradations. These samples received no print-and-capture degradation or only normalization. A random registration residual of approximately 0.7 px was also added before CNN input to simulate subpixel deviations after actual ECC fine registration. The resulting data covered ideal inputs, mild and compound degradations, and small registration residuals. This diversity helped the network learn a stable mapping from normalized inputs to block-level watermark labels. It also provided the basis for robust digital HVC extraction from halftone-dot features degraded by real print-and-capture channels.

2.4 CNN Watermark Decoder

2.4.1 Network Architecture

We designed an encoder-style fully convolutional network for block-level watermark decisions, as shown in Fig. 2. The network contained 7.11 million parameters. The 512×512 input associated each watermark pixel with a 16×16 carrier region. Four successive twofold downsampling operations reduced the feature map to 32×32 pixels. This resolution exactly matched the block-level watermark. The network therefore omitted U-Net-style upsampling and skip connections and performed watermark classification directly at the lowest resolution.

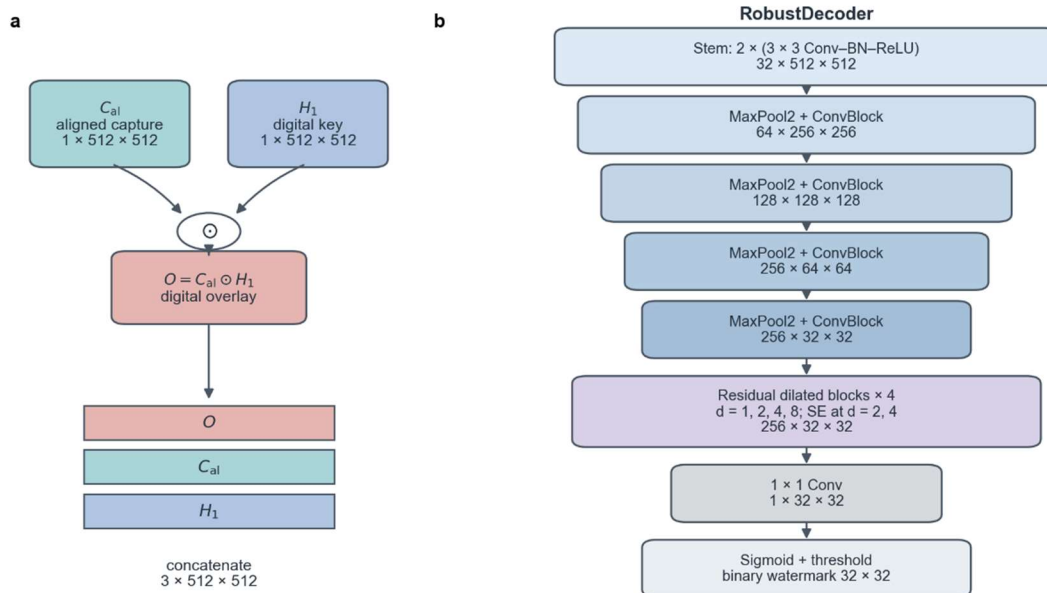


Figure 2. Three-channel input (a) and architecture of the CNN watermark decoder (b)

The network front end contained one stem convolutional block and four downsampling blocks. The stem expanded the input from 3 to 32 channels while preserving the 512×512 resolution. The four downsampling blocks then produced 64, 128, 256, and 256 channels at resolutions of 256×256 , 128×128 , 64×64 , and 32×32 . Each basic convolutional unit comprised two sequences of 3×3 convolution, batch normalization, and ReLU activation. Downsampling used 2×2 max pooling.

Four dilated residual blocks followed the 32×32 feature map. Each block had 256 channels, with dilation rates of 1, 2, 4, and 8. Dilated convolution expanded the receptive field without further reducing spatial resolution, allowing the network to aggregate screen-phase statistics from adjacent watermark blocks. The second and third blocks incorporated squeeze-and-excitation (SE) channel attention, which adaptively reweighted channel responses using global feature statistics. A final 1×1 convolution mapped the 256-channel features to single-channel 32×32 block-level logits.

This architecture has three advantages. First, the output resolution directly matches the block-level watermark structure and requires no pixel-level upsampling. Second, multiscale dilated residual blocks integrate cross-block context, improving recovery from local dot damage and broken watermark strokes. Third, SE attention emphasizes feature channels that are more sensitive to differences between the reference screen and the shifted screen.

2.4.2 Loss Function and Model Training

Watermark foreground pixels accounted for approximately 15% of all watermark blocks. Using binary cross-entropy alone could therefore bias the network toward predicting most blocks as background. We combined binary cross-entropy and Dice losses to balance blockwise classification accuracy with foreground completeness.

Let the predicted probability and ground-truth label of watermark block k be p_k and w_k , respectively. For $N = 32 \times 32$ blocks, the binary cross-entropy loss is

$$L_{\text{BCE}} = -\frac{1}{N} \sum_{k=1}^N [w_k \log p_k + (1 - w_k) \log (1 - p_k)] \quad (12)$$

The soft Dice loss is

$$L_{\text{Dice}} = 1 - \frac{2 \sum_{k=1}^N p_k w_k + \varepsilon}{\sum_{k=1}^N p_k + \sum_{k=1}^N w_k + \varepsilon} \quad (13)$$

The final loss function is

$$L = L_{\text{BCE}} + L_{\text{Dice}} \quad (14)$$

Here, L_{BCE} supervises the classification of each watermark block, while L_{Dice} directly constrains overlap between the predicted foreground and ground-truth watermark. This combination reduces all-background predictions caused by class imbalance.

The training procedure used three strategies.

(1) Progressive degradation learning. Mild print-and-capture degradations were applied during the first five epochs, allowing the network to learn basic screen differences in HVC watermarks. From epoch 6, the full-strength seven-stage channel was used to gradually adapt the network to complex compound distortions.

(2) Incorrect-key samples. For 25% of training samples, the correct key was replaced with an incorrect key H_1' generated by random translation, and the stacked image was recalculated as

$$O' = H_1' \odot C_{\text{al}} \quad (15)$$

The supervisory target was then set to an all-zero image. This strategy forced the network to suppress watermark output when the key was mismatched. It also prevented the CNN from bypassing the digital key and inferring the watermark solely from carrier content.

(3) Registration-residual simulation. Random translations within ± 0.7 px were added to finely registered samples. This expanded the training distribution to include residual errors after real ECC registration and improved tolerance to subpixel misalignment.

The network was optimized using AdamW with an initial learning rate of 1×10^{-3} . A cosine-annealing schedule adjusted the learning rate. The batch size was 4, and four gradient-accumulation steps produced an effective batch size of 16. Training continued for 40 epochs. Inference retained only geometric correction, subpixel registration, digital stacking, and CNN forward computation.

In summary, fiducial markers and digital-key registration first corrected the major geometric distortions introduced by printing and capture. The CNN decoder then compensated for dot spread, blur, illumination variation, noise, and other degradations that registration could not repair. The method thus replaced manual visualization using a physical key share with automated digital decoding.

3. Experiments and Results

3.1 Experimental Setup

3.1.1 Simulated and Real Data

We conducted simulated-degradation and real print-and-capture experiments to evaluate the proposed method under varied conditions. The simulated experiments used 5,000 natural carrier images from COCO val2017. At the original-image level, 4,000 images were assigned to training, 500 to validation, and 500 to testing. This split prevented degraded versions of the same carrier from appearing in both training and test sets. Training watermarks were generated randomly. The validation and test sets additionally included real logo patterns unseen during training to evaluate generalization to unknown watermark shapes. Samples were generated online as described in Section 2.3.2, with carrier images, watermark patterns, and channel parameters resampled at each iteration.

Real test sheets were produced on standard A4 copy paper using a laser printer at 300 dpi. Watermarked halftone images were printed at their original size with a 1:1 scale. Each sheet contained six 512×512 -pixel watermarked patches surrounded by ArUco fiducial markers. The digital key was not printed and remained only at the verification endpoint. A smartphone captured the sheets from different distances and angles, yielding 60 real print-and-capture images.

3.1.2 Experimental Environment and Training Parameters

Network training and testing used PyTorch 2.11 and CUDA 12.8 on four NVIDIA GeForce RTX 3060 GPUs, each with 12 GB of memory. The CNN decoder received a $512 \times 512 \times 3$ tensor comprising the digitally stacked image, finely registered captured image, and digital key. It produced 32×32 block-level logits, which the sigmoid function mapped to a watermark probability map.

The network used AdamW and cosine-annealing learning-rate scheduling with an initial learning rate of 1×10^{-3} . The batch size was 4, and four accumulation steps produced an effective batch size of 16. Training lasted 40 epochs. Mild degradations were used for the first five epochs, allowing the network to learn basic screen differences. From epoch 6, the full-strength seven-stage degradation model was applied to adapt the network to complex physical-channel distortions. During inference, sigmoid probabilities were binarized at a block-level threshold of 0.5.

3.1.3 Comparison Methods

We compared three extraction methods under real print-and-capture conditions. All methods used identical test samples, ECC registration results, and digital keys to ensure consistent comparison conditions.

(1) Direct print-and-capture stacking. The registered captured image C_{al} was multiplied by the digital key H_1 to obtain the stacked image

$$O = H_1 \cdot C_{al} \quad (16)$$

Watermark blocks were then classified using the thresholding rule described in Section 3.1.4.

(2) Post-ECC blockwise XOR extraction. The registered captured image was binarized and XORed with the digital key. Blockwise voting then produced the extracted watermark.

(3) Proposed CNN decoding. The stacked image O , registered image C_{al} , and digital key H_1 formed a three-channel input. The CNN produced a 32×32 block-level watermark prediction.

3.1.4 Evaluation Metrics for Watermark Extraction

We evaluated both registration and watermark decoding. Registration error was calculated as the mean Euclidean distance between corresponding corners before and after correction. This metric quantified geometric deviation between the captured image and the reference coordinates. Watermark extraction was evaluated using the block-level balanced correct decoding rate (CDR), reducing the effect of foreground-background class imbalance.

Watermark foreground blocks represented approximately 15% of all blocks. Overall accuracy could therefore remain high even if every block were classified as background, obscuring actual foreground recovery. We consequently used block-level balanced CDR as the primary metric:

$$CDR_{bal} = \frac{1}{2} \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \quad (17)$$

Here, TP , TN , FP , and FN denote true positives, true negatives, false positives, and false negatives in block-level predictions, respectively. The metric accounts for both watermark-foreground recall and background-recognition rate, with an expected random-decision value of approximately 50%. We also reported the median, minimum, and proportion of samples satisfying $CDR_{bal} \geq 75\%$.

3.2 Geometric Correction and Registration Results

Among the 60 real captured images, 57 contained at least two valid ArUco markers and underwent coarse homography correction. The remaining three images were excluded from registration and decoding because markers were occluded, blurred, or insufficiently detected. The localization-stage pass rate was therefore 95.0%.

Residual misalignment was measured as the mean Euclidean distance between corresponding corners in the registered and reference images. Table 1 summarizes the results. After coarse ArUco correction, the median residual misalignment remained 8.74 px. Fiducial markers and homography alone therefore could not provide the dot-level precision required for HVC stacking. Phase-correlation initialization followed by ECC fine registration reduced the median residual to 0.25 px, a decrease of approximately 97.1%. Residual misalignment did not exceed 0.87 px for 90% of samples. All 57 samples entering the ECC stage converged successfully.

These results show that phase correlation followed by ECC fine registration effectively removed residual translation, rotation, and local perspective errors after coarse correction. The method therefore provided subpixel alignment for subsequent watermark extraction. However, subpixel registration did not restore captured images to ideal digital halftones. Photometric degradations from ink spread, blur, illumination variation, and resampling remained. The registration experiment

confirmed control of major geometric errors, while dot-structure damage and photometric distortion remained the primary limitations on extraction performance.

Table 1. Residual misalignment and processing success rates during geometric correction and registration

Processing stage	Valid samples	Median residual misalignment (px)	90th-percentile residual misalignment (px)	Processing success rate
ArUco localization and coarse correction	57	8.74	—	57/60 (95.0%)
ECC registration	57	0.25	0.87	57/57 (100%)

3.3 CNN Decoding Results under Simulated Conditions

Figure 3 shows the 40-epoch training process and test-set performance of the CNN decoder. The training loss was 0.423 after the first epoch and decreased rapidly as the network learned screen-phase differences. When training switched from mild to full-strength seven-stage degradation at epoch 6, the loss briefly increased to 0.307, indicating greater decoding difficulty. It subsequently decreased and converged to 0.022 by epoch 40. Training and validation metrics followed similar trends, with no clear divergence indicating overfitting.

On the fixed test set, the CNN decoder achieved a block-level CDR of 99.42% under degradation-free digital conditions. Under compound degradation, the CDR remained 99.16%. The compound channel simultaneously applied ink spread, lens blur, homography, resampling, illumination and contrast changes, sensor noise, and JPEG compression. Performance decreased by only 0.26 percentage points from the degradation-free condition.

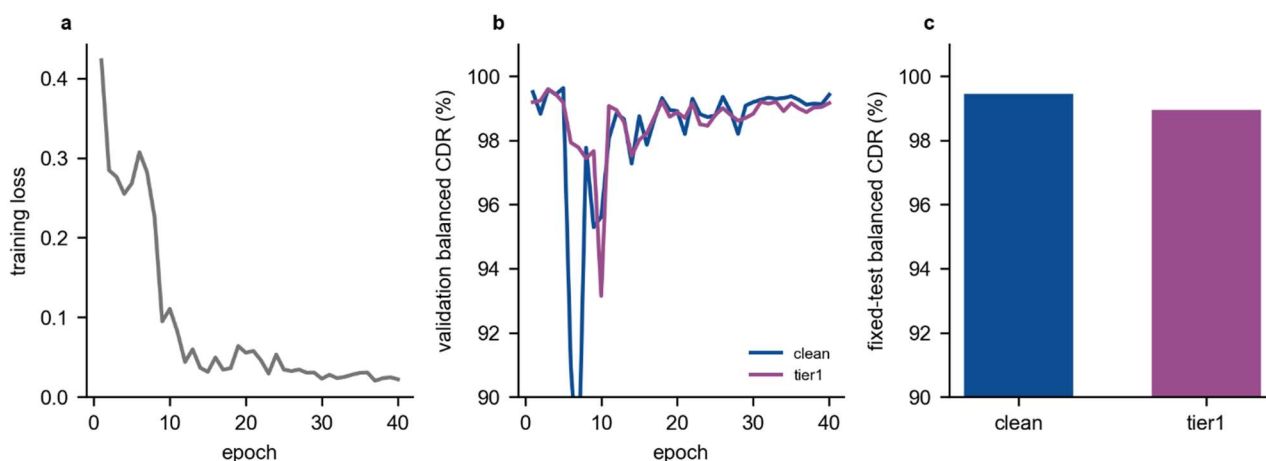


Figure 3. Training process and fixed-test-set performance of the CNN watermark decoder

(a) Training loss; (b) validation-set CDR under clean and degraded conditions; (c) fixed-test-set CDR under clean and degraded conditions

Within the specified channel-parameter ranges, online compound-degradation samples enabled the CNN to learn a stable watermark representation. The decoder therefore maintained high block-level performance under multiple simultaneous simulated distortions. However, this result represents performance only within the constructed simulation distribution. Adaptation to physical channels required further evaluation using real print-and-capture experiments.

3.4 Watermark Extraction under Real Print-and-Capture Conditions

3.4.1 Quantitative Comparison of Extraction Methods

Table 2 and Fig. 4 present the watermark-extraction performance of the three methods on real print-and-capture samples.

Table 2. Watermark-extraction performance under real print-and-capture conditions

Extraction method	Mean CDR (%)	Median CDR (%)	Minimum CDR (%)	Samples with CDR $\geq 75\%$ (%)
Direct stacking	53.5	49.0	39.3	0
XOR extraction	83.1	85.8	58.9	82
CNN decoding	96.9	98.8	72.7	98

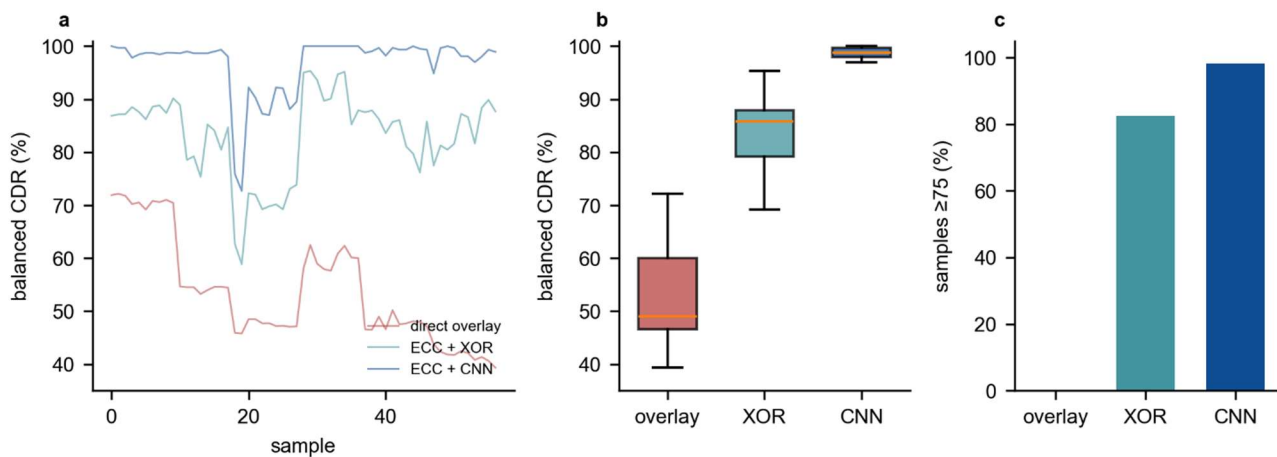


Figure 4. Performance comparison of watermark-extraction methods under real print-and-capture conditions

(a) CDR of each method; (b) CDR distributions; (c) proportion of samples with CDR $\geq 75\%$

Direct digital stacking achieved a mean CDR of 53.5%, a median of 49.0%, and a minimum of 39.3%. No sample reached a CDR of 75%. Because random binary decisions yield an expected CDR of approximately 50%, direct stacking remained unable to recover watermarks reliably after subpixel registration under real conditions.

Blockwise XOR extraction increased the mean CDR to 83.1%, with a median of 85.8% and a minimum of 58.9%. Samples with CDR values of at least 75% accounted for 82%. This method partly reduced the effect of continuous grayscale variation on watermark decisions, although performance still varied considerably among samples.

The proposed CNN achieved a mean CDR of 96.9%, a median of 98.8%, and a minimum of 72.7%. Samples with CDR values of at least 75% accounted for 98%. The mean CDR exceeded direct digital stacking by 43.4 percentage points and post-ECC blockwise XOR extraction by 13.8 percentage points. The CNN therefore improved both average decoding performance and stability across real captured samples.

3.4.2 Visual Analysis of Watermark Extraction

Figure 5 compares visual extraction results. Directly stacked images retained strong halftone-dot textures, while density differences between watermark and background regions remained weak. Uneven backgrounds obscured fine strokes and caused blurred edges and local structural breaks. Blockwise XOR strengthened responses in some watermark regions but produced many isolated

erroneous blocks. Fine strokes and internal regions also lacked continuity. In contrast, CNN outputs more closely matched the ground-truth watermarks in overall contour, foreground distribution, and internal structure. Across all six samples, the CNN removed most carrier textures and isolated errors, separating watermark regions from complex backgrounds. It also preserved connectivity and structural integrity for watermarks with fine branches, enclosed regions, or irregular boundaries. Compared with blockwise XOR, the CNN substantially reduced speckle noise, produced more continuous edges, and partially repaired local defects and internal holes. The decoder therefore did more than enhance watermark contrast. Its multichannel input enabled learning of a nonlinear mapping between degraded halftone dots and watermark blocks, supporting both background suppression and local-structure recovery.

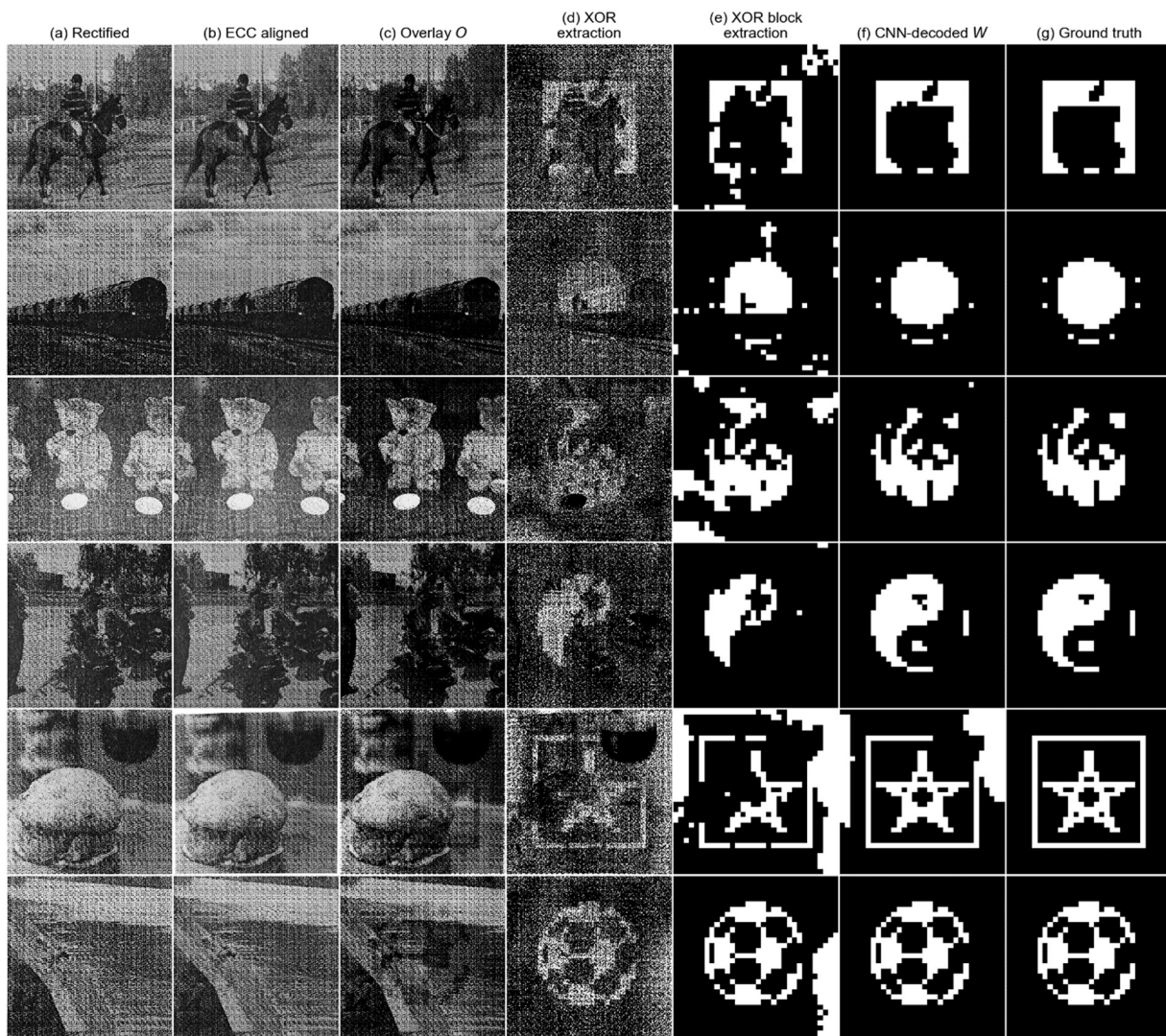


Figure 5. Extraction results for six real samples

4. Conclusion

We proposed an HVC watermark extraction method that combines a print-and-capture degradation model with a CNN decoder. The method addresses dot-structure damage and photometric distortion that prevent complete and clear digital extraction. A seven-stage model generated training samples with ink spread, lens blur, geometric distortion, resampling, illumination and contrast variation, sensor noise, and JPEG compression. The CNN used the digitally stacked image as its primary input and the registered image and digital key as auxiliary inputs to recover 32×32 block-level watermarks.

The decoder achieved a CDR of 99.16% under simulated compound degradation and a mean CDR of 96.9% on real print-and-capture samples. These values exceeded direct stacking and post-ECC blockwise XOR extraction by 43.4 and 13.8 percentage points, respectively. The results indicate that the decoder learned a nonlinear mapping from damaged halftone-dot features to watermarks. This approach provides a feasible basis for mobile digital extraction of HVC watermarks under the evaluated print-and-capture conditions.

References

- [1] Zou, K. C. (2021). *Research on digital image watermarking technology* [Master's thesis]. Xidian University. <https://doi.org/10.27389/d.cnki.gxadu.2021.002305>
- [2] Wang, Y. Y. (2021). *Research on image encryption and its security based on visual cryptography* [Master's thesis]. Xi'an University of Technology. <https://doi.org/10.27398/d.cnki.gxalu.2021.000475>
- [3] Guo, J. M., & Seshathiri, S. (2023). Visually encrypted watermarking for ordered-dithered clustered-dot halftones. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(8), 4375–4387. <https://doi.org/10.1109/TCSVT.2023.3237688>
- [4] Fu, M. S., & Au, O. C. (2002). Data hiding watermarking for halftone images. *IEEE Transactions on Image Processing*, 11(4), 477–484. <https://doi.org/10.1109/TIP.2002.999680>
- [5] Guo, Y. F., Au, O. C., Wang, R., & et al. (2018). Halftone image watermarking by content aware double-sided embedding error diffusion. *IEEE Transactions on Image Processing*, 27(7), 3387–3402. <https://doi.org/10.1109/TIP.2018.2815181>
- [6] Wang, Y. Y., Ni, R. R., Zhao, Y., & Xian, M. (2018). Watermark embedding for direct binary searched halftone images by adopting visual cryptography. *Computers, Materials & Continua*, 55(2), 255–265. <https://doi.org/10.3970/cmc.2018.01732>
- [7] Rosen, J., & Javidi, B. (2001). Hidden images in halftone pictures. *Applied Optics*, 40(20), 3346–3351. <https://doi.org/10.1364/AO.40.003346>
- [8] Basak, O., & Gaurav, S. (2009). Continuous phase-modulated halftones. *IEEE Transactions on Image Processing*, 18(12), 2718–2734. <https://doi.org/10.1109/TIP.2009.2028367>
- [9] Guo, J.-M., & Seshathiri, S. (2019). Watermarking in dot-diffusion halftones using adaptive class-matrix and error diffusion. *ECTI-CIT Transactions*, 13(1), 1–8.
- [10] Oztan, B., & Sharma, G. (2010). Multiplexed clustered-dot halftone watermarks using bi-directional phase modulation and detection. In *2010 IEEE International Conference on Image Processing* (pp. 981–984). IEEE. <https://doi.org/10.1109/ICIP.2010.5651620>
- [11] Oztan, B., & Sharma, G. (2010). Per-separation clustered-dot color halftone watermarks: Separation estimation based on spatial frequency content. *Journal of Electronic Imaging*, 19(4), Article 043007. <https://doi.org/10.1117/1.3497615>
- [12] Ulichney, R., Pollard, S., & Gaubatz, M. (2014). Fast mobile stegatone detection using the frequency domain. In *NIP & Digital Fabrication Conference* (Vol. 30, No. 1, pp. 237–242). Society for Imaging Science and Technology. https://doi.org/10.2352/ISSN.2169-4451.2014.30.1.art00059_1
- [13] Tancik, M., Mildenhall, B., & Ng, R. (2020). StegaStamp: Invisible hyperlinks in physical photographs. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2114–2123). IEEE. <https://doi.org/10.1109/CVPR42600.2020.00219>
- [14] Fang, H., Zhang, W. M., Zhou, H., & et al. (2019). Screen-shooting resilient watermarking. *IEEE Transactions on Information Forensics and Security*, 14(6), 1403–1418. <https://doi.org/10.1109/TIFS.2018.2878541>
- [15] Qin, C., Li, X. M., Zhang, Z. Y., & et al. (2024). Print-camera resistant image watermarking with deep noise simulation and constrained learning. *IEEE Transactions on Multimedia*, 26, 2164–2177. <https://doi.org/10.1109/TMM.2023.3293272>
- [16] Ma, Z. H., Fang, H., Yang, X., & et al. (2025). RoPaSS: Robust watermarking for partial screen-shooting scenarios. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 18, pp. 19332–19339). <https://doi.org/10.1609/aaai.v39i18.34128>