Reconstruction of the image captured by TempestSDR

Filip Tuch, Richard Ostertág[*]

Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia

Abstract

A computer's image is sent to the monitor as a stream of RGB pixels, with their intensities encoded as voltage levels in the signal traveling through the cable. During transmission, these signals give off unwanted electromagnetic emissions, which can be picked up using TempestSDR software to reconstruct the original image. However, the resulting image is often noisy and hard to read. With standard software defined radios (SDR), you can only eavesdrop on the image from a short distance, about 50 cm. In this paper, we propose two methods to improve the quality of the captured image. First, we used a directional Yagi-Uda antenna, a signal amplifier, and a band-pass filter. This setup extended the capture distance to around 20 meters. Even so, the image remained noisy, so we turned to deep learning for our second approach. We created scripts to automate TempestSDR and the dataset generation process, producing a large dataset covering typical everyday computer use. We selected two convolutional neural networks designed for image reconstruction, trained them on our dataset, and evaluated their performance using standard image quality metrics. The results showed a significant improvement in reconstructed image quality across all tested metrics. Finally we integrated the trained models into TempestSDR, making high-quality image reconstruction much easier for users. Our findings highlight potential vulnerability of display devices and emphasize the need for preventive measures to enhance their security.

Keywords

electromagnetic side-channel attack, TempestSDR, convolutional neural network, image processing

1. Introduction

Confidentiality and privacy are fundamental pillars of information security. While conventional approaches focus primarily on mechanisms like authentication, encryption, and access control, hardware-based vulnerabilities receive considerably less attention. Among these, side-channel attacks exploit unintended information leakage arising from standard hardware operations.

Electromagnetic (EM) emissions have long been recognized as a source of interference in radio communications. Historically, most efforts focused on minimizing such emissions to prevent signal degradation. However, military and intelligence institutions also recognized a more critical implication: the possibility of intercepting these emissions to reconstruct sensitive data. Because early research on this topic remained classified [1], it contributed to a widespread perception, that such attacks required specialized, expensive equipment and were thus impractical for the general public. As a result, consumer electronics were rarely designed with protection against side-channel attacks in mind.

Computer monitors (and display devices in general) present a particularly viable target in this context. During image transmission from a computer to a monitor, unintentional EM emissions are inadvertently produced. These emissions can be intercepted and analyzed to reconstruct the original screen content. The first public demonstration of such an attack was conducted by Wim van Eck in 1985, who successfully recovered images from analogue CRT monitors [2]. Although LCD monitors and modern digital interfaces were initially believed to be immune to this threat, Markus Kuhn demonstrated in 2005 that digital systems can, in fact, be even more susceptible [3].

Today, this type of attack can be executed using TempestSDR [4], a software-defined radio (SDR) application, capable of reconstructing images from captured EM emissions in real time. However, using standard equipment (such as a common omnidirectional antenna), our measurements showed, that the

ITAT'25: Information Technologies – Applications and Theory, September 26–30, 2025, Telgárt, Slovakia
[*]Corresponding author.
tuch1@uniba.sk (F. Tuch); richard.ostertag@fmph.uniba.sk (R. Ostertág)
0000-0002-6560-1515 (R. Ostertág)

effective range for a successful image reconstruction is limited to under 50 centimeters. Furthermore, the reconstructed images are typically significantly compromised by noise and frequently lack clarity.

This paper investigates methods for extending the effective attack range and improving the quality of reconstructed images. The following section outlines the principles of image transmission, its inherent vulnerabilities, and explains how TempestSDR processes EM emissions to recover image content. The subsequent sections present two methods for improving both the range and visual clarity of the reconstructed output.

2. Video interfaces and their vulnerabilities

Several video interfaces are used to transmit image data from a computer to a display device. These include both analogue and digital standards, each of which operates in a way that makes them susceptible to EM leakage.

VGA interface. Video Graphics Array (VGA) [5] is an analogue display interface, that transmits image data using separate voltage signals for the red, green, and blue (RGB) channels. Each color component is encoded as an analogue voltage signal ranging from 0 V to 0.7 V. In addition, two digital synchronization signals, HSync and VSync (typically 0 V or 5 V), are used to coordinate horizontal and vertical refresh cycles.

The video signal is transmitted as a continuous stream of pixels. These pixels form horizontal scan lines, with each scan line ending in a short blanking interval marked by the HSync signal. Once all scan lines for a frame have been transmitted, a vertical blanking follows, indicated by the VSync signal. This process repeats continuously, producing a sequence of full-screen image frames.

HDMI interface. High-Definition Multimedia Interface (HDMI) [6] is a digital alternative to analogue interfaces such as VGA. Unlike VGA, which transmits pixel intensities through analogue voltage levels, HDMI transmits data serially using Transition-Minimized Differential Signaling (TMDS). The interface uses four TMDS channels, one for each RGB channel, and one for clock signals.

Each pixel's color components are represented as 8-bit values, which are converted into 10-bit TMDS symbols. These symbols are then transmitted as a high-speed serial data stream. Like VGA, the stream of pixels is organized into horizontal scan lines, which in turn form complete image frames. Synchronization signals are embedded within the stream, analogous to HSync and VSync.

EM susceptibility. The core principle behind the attacks demonstrated by Van Eck and Kuhn is that image transmission relies on rapidly changing voltages to represent pixel colour intensities [2, 3]. These voltage transitions emit EM radiation in the VHF (from 30 MHz to 300 MHz) and UHF (from 300 MHz to 3 GHz) radio bands.

When analyzing captured emissions from analogue standards, TempestSDR leverages the fact that image data is transmitted at a rate defined by the monitor's pixel clock frequency f_m, which is the product of display width x_t, height y_t, and frame rate f_v. Then, the time required to transmit a single pixel is

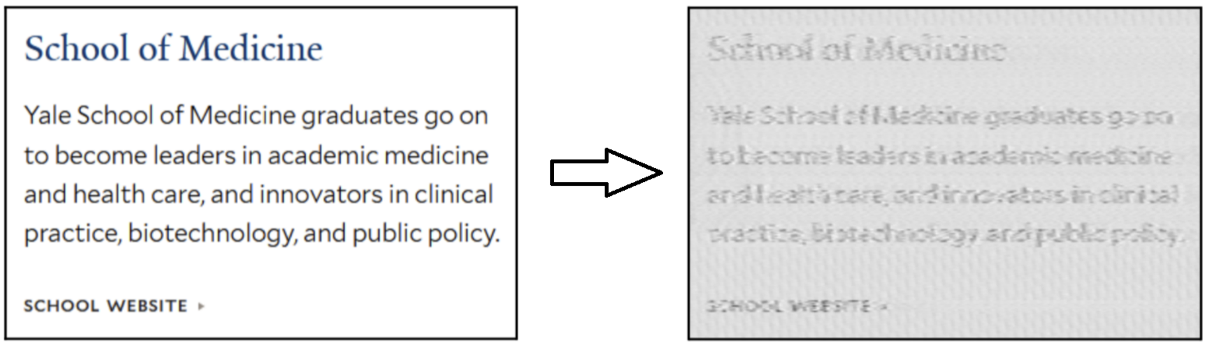
$$t_p = 1/(x_t \cdot y_t \cdot f_v), \tag{1}$$

which results in impulses in the captured signal occurring at multiples of t_p. In essence, the amplitude of each impulse corresponds to the intensity of the pixel being transmitted at that moment. As a result, the signal can be properly segmented according to pixel transmission periods, and the amplitude of each impulse can be used to infer the approximate grayscale value for the corresponding pixel.

For digital transmission, where each pixel intensity is represented using n_b bits, the bit period is

$$t_b = 1/(x_t \cdot y_t \cdot f_v \cdot n_b). \tag{2}$$

TempestSDR reconstructs the image by averaging the bit-level signal over each pixel period. Note that for digital transmissions, in which Display Stream Compression (DSC) is used, e.g. HDMI 2.1, an


School of Medicine

Yale School of Medicine graduates go on to become leaders in academic medicine and health care, and innovators in clinical practice, biotechnology, and public policy.

SCHOOL WEBSITE ▸

Figure 1: A close-up view of an attack in TempestSDR (from left to right: original image, reconstructed image)

attack with the described approach is not feasible. However, it may still be possible to eavesdrop on the computer screen itself, rather than the cable.

Beyond signal demodulation, TempestSDR also offers automatic detection of the target monitor's resolution and refresh rate, allowing relatively straightforward attacks even in cases where these parameters are initially unknown.

Baseline attack. An illustrative eavesdropping attempt involves a VGA monitor with a resolution of 1024×768 pixels operating at a 60 Hz refresh rate. Using a USRP x310 software-defined radio, the emissions were captured at a distance of approximately 50 centimeters. TempestSDR was tuned to the second harmonic frequency of the monitor's pixel clock, which, in this case, is approximately 130 MHz. A close-up comparison of the original (reference) image and the reconstructed output obtained through this baseline setup is shown in Figure 1.

3. Extending the range of attack

Although much attention is typically given to software-based processes such as signal demodulation, the ability to receive a clean and sufficiently strong signal at a distance is equally important. In electromagnetic (EM) side-channel attacks, this makes hardware optimization a critical factor.

Without appropriate hardware, the effective capture range is limited to very short distances (approximately 50 centimeters in our baseline setup). To extend this range, we employ a high-gain antenna, a low-noise amplifier, and a band-pass filter. This setup not only increases the reception range, but also improves the signal-to-noise ratio, thereby enhancing the quality of the reconstructed images.

Directional antennas. The most common types of directional antennas applicable for this scenario are the log-periodic antenna and the Yagi-Uda antenna. While both are directional, they differ in bandwidth and frequency gain. Log-periodic antennas typically cover a wider frequency range (e.g. from 100 MHz to 1 GHz), whereas Yagi-Uda antennas have a narrow bandwidth, around 3% from their centre frequency. On the other side, Yagi-Uda antennas commonly offer a higher gain compared to log-periodic antennas.

Log-periodic antennas are more suitable when the frequency of interest is not known in advance, making them ideal for exploratory attacks. In contrast, the Yagi-Uda antenna is optimal when the monitor specifications are known, allowing the antenna to be tuned precisely to the required frequency for maximum signal gain. Previous works also support this rationale: van Eck originally used a Yagi-Uda antenna [2], while Kuhn employed both [3]. More recently, Meulemeester demonstrated a successful image reconstruction from a distance of 80 meters using two professional high-gain log-periodic antennas, double amplification, and band-pass filtering [7].

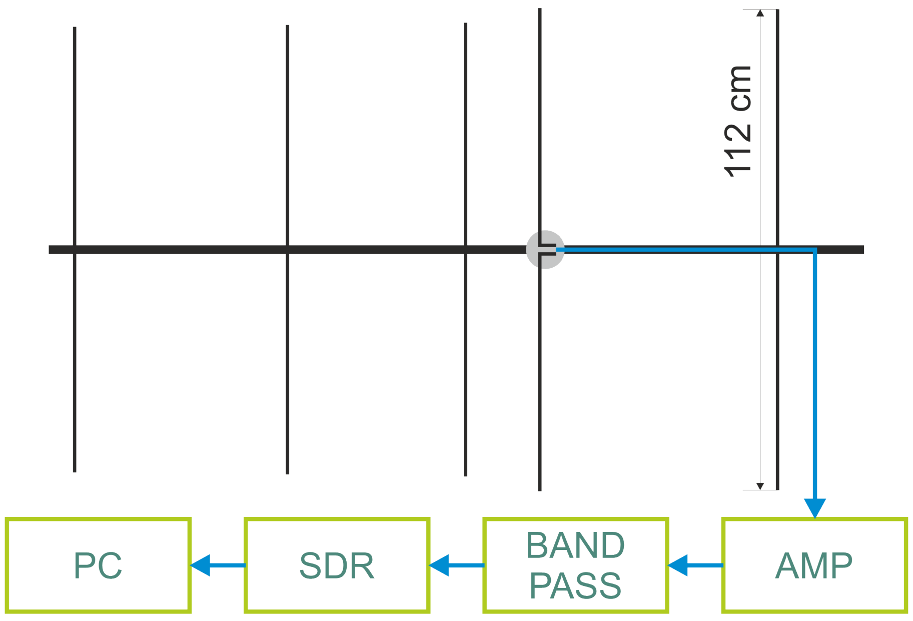


Figure 2: Our Yagi-Uda system (with antenna to scale)

Our Yagi-Uda implementation. Given our knowledge of the target monitor's characteristics, we designed a 5-element Yagi-Uda antenna with approximately 8 dBd gain tuned to the second harmonic frequency of the monitor's pixel clock (approximately 130 MHz). The antenna geometry was calculated using an online Yagi-Uda calculator from Changpuak [8]. We decided to build the antenna from aluminium for lightness and compactness. Both straight and folded dipole variants were implemented.

The antenna was connected to the SDR using a 50 Ω coaxial cable. Since the folded dipole has a higher characteristic impedance (typically from 200 Ω to 300 Ω), we implemented a 4:1 balun to match it to the SDR's 50 Ω input. A TQP3M9037 low-noise amplifier was added in line, providing an additional 20 dB of gain. Using a Tektronix MDO4104C multi-domain oscilloscope, we observed substantial interference in the 90-120 MHz band, mostly from FM radio broadcasts. To protect the SDR's input channel and to isolate the desired signal, we applied a band-pass filter tuned to 125-135 MHz. A schematic setup is shown in Figure 2.

Results. With inspiration from [7], we created a reference image (see Figure 3) incorporating diminishing text and standard image quality symbols, including SMPTE color bars, a Tuning Signals test card, and a checkerboard pattern.

We tested the implemented system in a hallway with a metal structure, containing various reflective and absorptive objects. The target device, a laptop connected to a monitor via an HDMI-to-VGA converter, and a basic unshielded VGA cable, was positioned at one end of the hallway. The attacker remained in their room, while the Yagi-Uda antenna was placed in the hallway to capture emissions from the target. The test scenario is depicted in Figure 4, and reconstruction results from various distances are shown in Figure 5. Both folded and straight dipoles achieved similar results.

4. Recovering images with deep learning

While using optimized hardware allowed us to capture clearer signals from longer distances, the reconstructed images often remained degraded due to noise and distortion. To address this issue, we explored the application of deep learning techniques, specifically convolutional neural networks (CNNs), for image post-processing and enhancement.

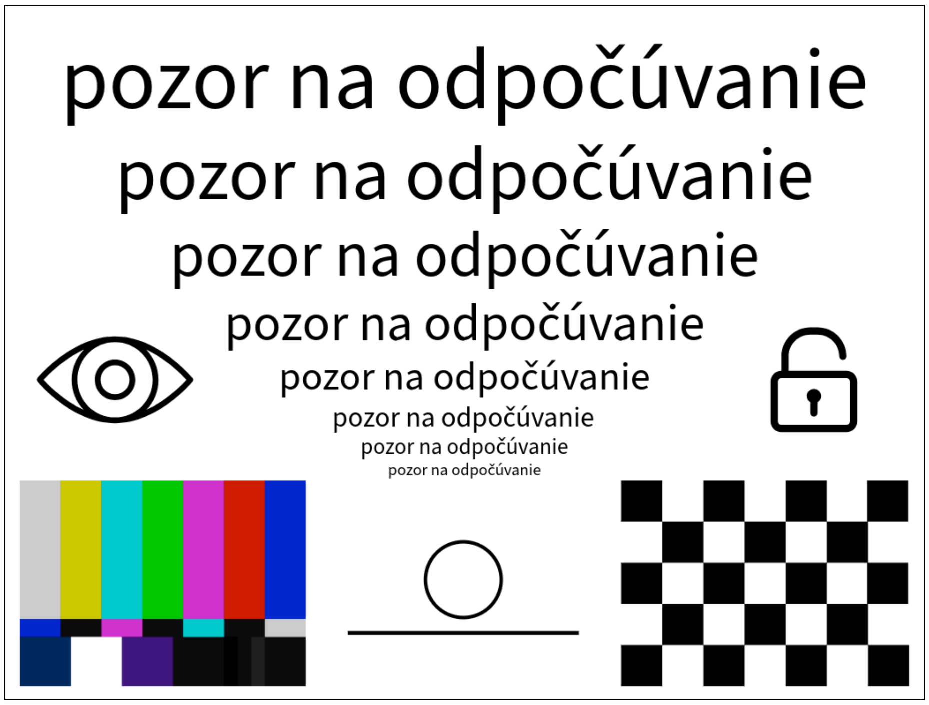


Figure 3: Reference image used for testing the implemented system

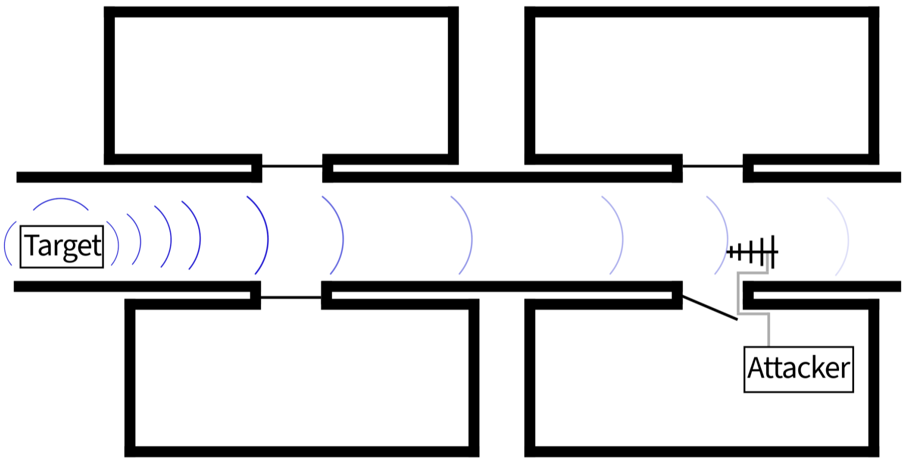


Figure 4: Attack scenario

Problem definition. The image restoration task can be formally formulated as recovering a clean image x from a degraded observation y, defined by the relationship $y = x + v$, where v represents degradation, e.g. noise.

In the context of electromagnetic side-channel attacks, previous works have demonstrated the feasibility of using deep learning to restore screen content from captured emissions [9, 10]. However, these works often relied on artificially generated low-resolution patches, which may not accurately reflect the challenges present in real-world attacks.

Our approach instead focuses on reconstructing entire frames obtained from actual TempestSDR captures. A similar objective was pursued in the Deep-Tempest paper [11], which also attempted to restore complete images from side-channel attacks. However, their method mainly processed complex-valued IQ (in-phase and quadrature) images captured using gr-tempest software[1], whereas we rely solely on amplitude images extracted from TempestSDR. With this approach, we try to make the attack

[1]TempestSDR reimplementation in GNU Radio.

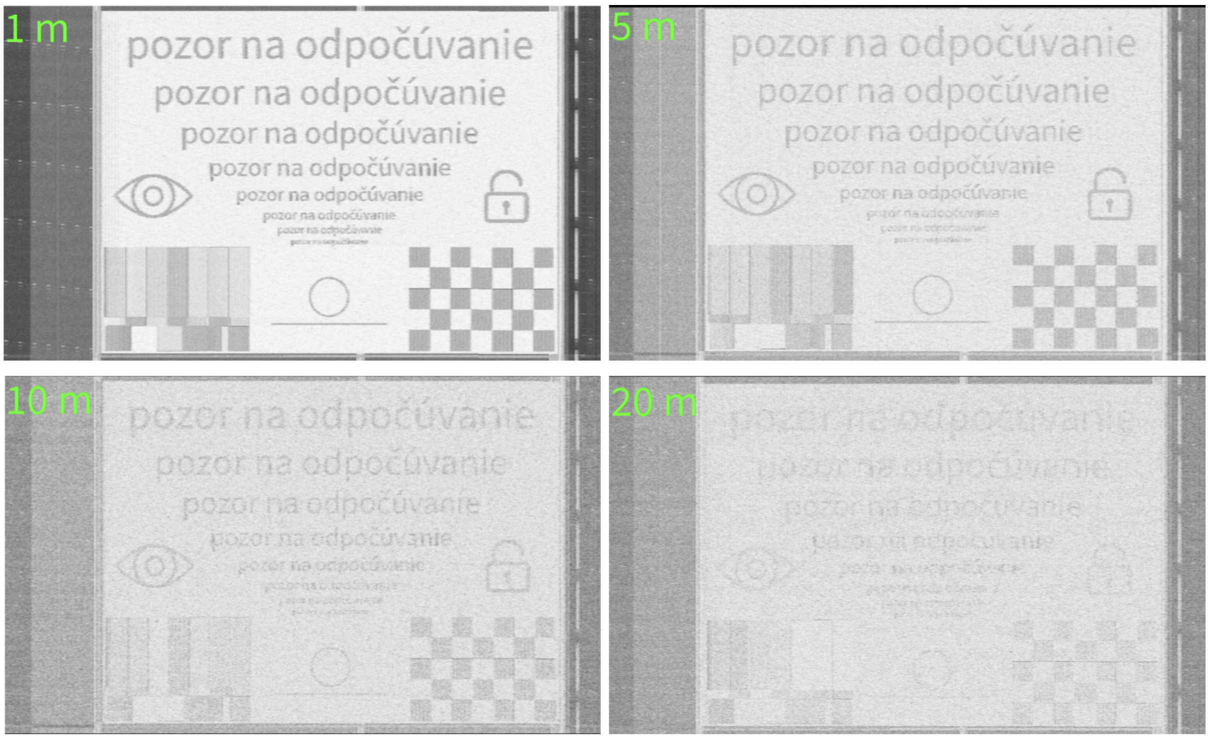


Figure 5: Captured image from various distances

more suitable for practical, real-time attack scenarios.

DnCNN architecture. DnCNN [12] is a CNN specifically designed for image denoising. Unlike traditional models that attempt to predict a clean image directly from a noisy input, DnCNN is trained to estimate the residual image, that is, the difference between the noisy and the clean image. This strategy simplifies the task by focusing the model on removing the noise component rather than reconstructing the entire image from scratch.

Let the noisy input image be $y = x + v$, where x is the clean image and v is additive noise. The model learns to approximate the residual mapping $\mathcal{R}(y) \approx v$, such that the clean image can be recovered via $x = y - \mathcal{R}(y)$. The network parameters Θ are optimized by minimizing the mean squared error (MSE) between the desired residual images and the estimated ones from the noisy input:

$$\ell(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \| \mathcal{R}(y_i; \Theta) - (y_i - x_i) \|^2, \tag{3}$$

where $(y_i, x_i)_{i=1}^{N}$ denotes a set of N noisy-clean image pairs [12].

The architecture uses a deep convolutional structure without pooling layers. DnCNN with the depth D consists of:

- Conv + ReLU (first layer) – 64 convolutional filters with the size of $3 \times 3 \times c$, where c is the number of input channels,
- Conv + BN + ReLU (hidden layers 2 to $D - 1$) – 64 convolutional filters with the size of $3 \times 3 \times 64$, with batch normalization between the convolution and activation function,
- Conv (final layer) – c filters with the size of $3 \times 3 \times 64$.

We selected this network for its simplicity, relatively low parameter count ($0.5 \cdot 10^6$ for $D = 17$ layers), and its effectiveness.

DRUNet architecture. DRUNet [13] is a flexible and high-performing CNN designed for various image restoration tasks, including denoising, deblurring, and super-resolution.

DRUNet formulates the task as an optimization problem, where the goal is to obtain the clean image x from the degraded image $y = \mathcal{T}(x) + v$, where $\mathcal{T}$ represents degradation independent of noise, and v is noise [13].

From a Bayesian perspective, DRUNet predicts the approximation $\hat{x}$ of the clean image x by solving the Maximum A Posteriori (MAP) problem, which is given as:

$$\hat{x} = \arg\max_{x} \log p(y|x) + \log p(x), \tag{4}$$

where $\log p(y|x)$ represents the log-probability of the observation of y with the given x, $\log p(x)$ represents a prior probability of the clean image, independent of the observation y. Formally, we can rewrite (4) as a minimization of the function:

$$\hat{x} = \arg\min_{x} \frac{1}{2\sigma^2} \parallel y - \mathcal{T}(x) \parallel^2 + \lambda \mathcal{R}(x), \tag{5}$$

where $\frac{1}{2\sigma^2} \parallel y - \mathcal{T}(x) \parallel^2$ is the data term, which ensures that the solution respects degradation, while $\lambda\mathcal{R}(x)$ is the regularization term, which enforces the desired properties on the solution [13].

DRUNet's architecture is based on the U-Net model, which features an encoder-decoder structure with skip connections. Each encoder stage consists of convolutional layers that downsample the input and extract features. The decoder stages then upsample these features to reconstruct the original image. DRUNet consists of a four-layer decoder (and encoder) with channel depths of 64, 128, 256, and 512, respectively. Each stage includes four residual blocks with the ReLU activation function, and downsampling/upsampling is performed via strided/transposed convolutions [13].

DRUNet normally expects both a degraded image and a noise level map as its input. Since the exact noise profile in our case is unknown, the noise map channel is set to zero.

We selected DRUNet for its strong denoising performance on various benchmark datasets [13], and for its proven use in similar applications such as Deep-Tempest [11].

Dataset creation. To simulate realistic screen content, we manually collected 650 screenshots of common desktop environments, popular websites, and various graphical user interfaces. Since improving text legibility is a primary objective of this work, we extended the dataset with 200 synthetically generated images containing randomly rendered text. These were created using a custom Python script built on the Selenium framework, which randomly selects font styles and sizes, renders the text in a browser window, and captures the resulting screen.

We fully automated the data acquisition process in TempestSDR using the PyAutoGUI library. A custom script was developed to iteratively display each reference image on the target monitor and trigger a snapshot in TempestSDR. For every reference image, eight captures were performed, each from a different distance and antenna orientation. The captures include images reconstructed from VGA and HDMI cable, as well as from HDMI-VGA converter emissions. These variations simulate real-world conditions, as even small changes in antenna placement can significantly affect the output image.

The snapshots produced in TempestSDR contain not only the reconstructed image but also padding caused by synchronization signals present during image transmission. Since the selected CNNs operate on pixel-wise comparisons between reference and captured images, this padding had to be removed. To accomplish this, we implemented a script that uses template matching from the OpenCV library. If the script failed to locate the reference image within the snapshot, the snapshot was deemed invalid and excluded from the dataset. Otherwise, the padding was removed, and the resulting image was retained for training.

Finally, the dataset was divided into training, testing, and validation subsets in a ratio of 80:10:10. The ratio of images captured from HDMI, VGA cable, and HDMI-VGA converter was preserved in the resulting subsets.

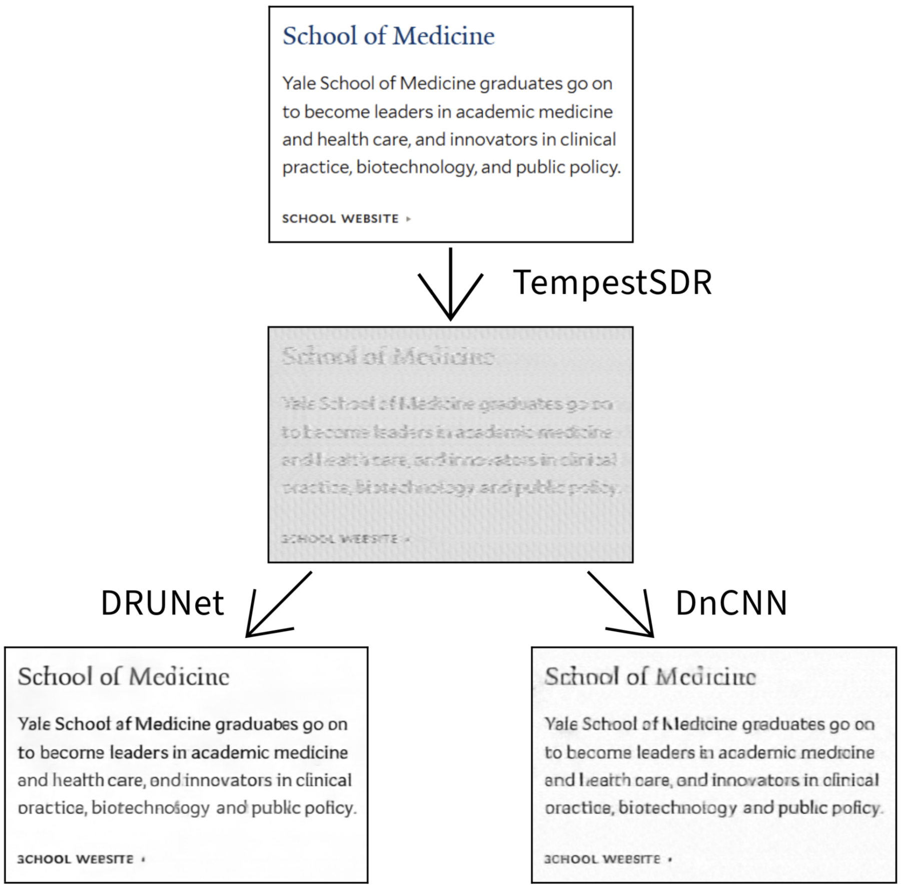
Table 1
Comparison of images reconstructed using only TempestSDR versus those reconstructed with the addition of trained models (on the testing dataset)

Method	PSNR	MSE	SSIM	CER	Reconstruction time
TempestSDR	8.65 dB	110.73	0.41	86 %	real time
DRUNet @ 70 epoch	19.08 dB	37.57	0.87	46 %	0.55 s
DnCNN @ 70 epoch	17.63 dB	50.16	0.75	54 %	0.25 s

Training. Both DnCNN and DRUNet were trained on the training subset with the goal of minimizing the MSE loss function. The initial learning rate was set to 10^{-4}, and the Adam Optimizer was employed. The input image pairs were split into patches of size 256×256 pixels by the dataloader, and the batch size was set to 32. Each model was trained for 100 epochs, with checkpoints saved every 10 epochs.

Results. After training the models, we evaluated the performance with standard image quality metrics: Peak Signal-to-Noise Ratio (PSNR), Mean Squared Error (MSE), and Structural Similarity Index Measure (SSIM). To better assess text readability, we additionally measured the Character Error Rate (CER), defined as the ratio of incorrectly recognised characters to the total number of characters in the reference image. We also recorded the average reconstruction time per image.

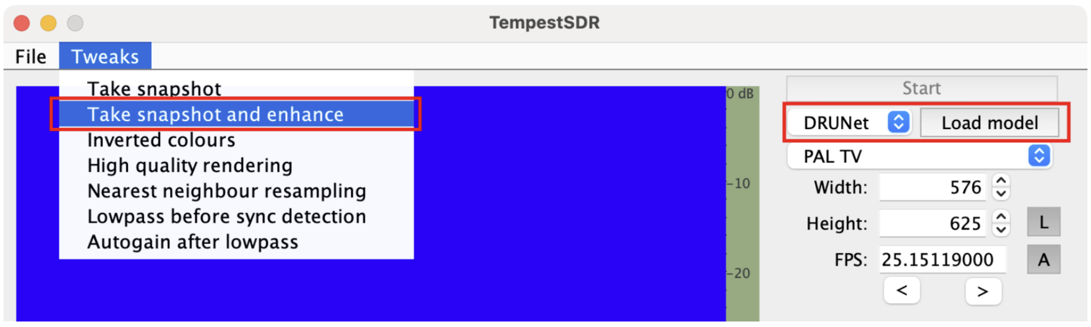
From the results in Table 1, it is evident that using DRUNet significantly enhances the quality of reconstructed images. The CER was reduced by 40%, greatly enhancing text readability in the images. While DnCNN also outperforms TempestSDR, it lags behind DRUNet across all image quality metrics. Its primary advantage is faster inference, requiring only half the reconstruction time compared to DRUNet. A real-world example of image reconstruction using both trained models is shown in Figure 6.


School of Medicine

Yale School of Medicine graduates go on to become leaders in academic medicine and health care, and innovators in clinical practice, biotechnology, and public policy.

SCHOOL WEBSITE ▸

Figure 6: Real world close-up example of image reconstruction using TempestSDR and trained models



DRUNet

PAL TV

Width: 576

Height: 625

FPS: 25.15119000

Figure 7: Added functionality in TempestSDR which enables the use of trained models during attacks

Integration into TempestSDR. Currently, in order to perform an attack using the trained models, the user has to manually capture an image in TempestSDR, remove padding from the captured image, run a separate script for image enhancement using a trained model, and then save the improved result. To streamline and simplify the attack process, we integrated a model inference functionality directly into TempestSDR.

Two new buttons were added to the GUI: Select Model (DRUNet or DnCNN) and Load Model (to specify the path to the trained model). Once loaded, the user can utilize the Take snapshot and enhance option to automatically enhance the captured image using the selected model. While removing the padding present in TempestSDR snapshots, contour detection is used to locate the region corresponding to the screen content. The added functionality can be seen in Figure 7.

5. Conclusion

We demonstrated that electromagnetic side-channel attacks on monitors, previously limited by short range and poor image quality, can be greatly improved through non-expensive hardware and software enhancements. By designing a directional Yagi-Uda antenna and applying a band-pass filter and amplifier, we extended the effective capture distance from 50 cm to 20 m. To further improve image quality, we trained CNNs on real TempestSDR captures, achieving up to a 40% improvement in text recognition accuracy. Finally, we integrated the models into TempestSDR, enabling practical and repeatable attacks with one-click image restoration.

Acknowledgments

This publication is the result of support under the Operational Program Integrated Infrastructure for the project: Advancing University Capacity and Competence in Research, Development and Innovation (ACCORD, ITMS2014+:313021X329), co-financed by the European Regional Development Fund.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-4o for grammar and spell check. After using the tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] National Security Agency, TEMPEST: A Signal Problem, https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/cryptologic-spectrum/tempest.pdf, 1972. Unclassified paper.

[2] W. van Eck, Electromagnetic radiation from video display units: An eavesdropping risk?, Computers & Security 4 (1985) 269–286. doi:10.1016/0167-4048(85)90046-X.

[3] M. G. Kuhn, Electromagnetic Eavesdropping Risks of Flat-Panel Displays, in: Privacy Enhancing Technologies, Springer Berlin Heidelberg, 2005, pp. 88–107. doi:10.1007/11423409_7.

[4] M. Marinov, Remote video eavesdropping using a software-defined radio platform, GitHub repository of TempestSDR software, 2014. URL: https://github.com/martinmarinov/TempestSDR.

[5] IBM VGA XGA Technical Reference, IBM, 1992. Manual.

[6] High-Definition Multimedia Interface Specification, HDMI Licensing, LLC, 2006. Manual.

[7] P. de Meulemeester, B. Scheers, G. A. E. Vandenbosch, Eavesdropping a (Ultra-)High-Definition Video Display from an 80 Meter Distance Under Realistic Circumstances, in: 2020 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI), 2020, pp. 517–522. doi:10.1109/EMCSI38923.2020.9191457.

[8] A. Changpuak, Yagi-Uda Antenna Calculator, https://www.changpuak.ch/electronics/yagi_uda_antenna_DL6WU.php, 2014.

[9] J. Galvis, S. Morales, C. Kasmi, F. Vega, Denoising of Video Frames Resulting From Video Interface Leakage Using Deep Learning for Efficient Optical Character Recognition, IEEE Letters on Electromagnetic Compatibility Practice and Applications 3 (2021) 82–86. doi:10.1109/LEMCPA.2021.3073663.

[10] F. Lemarchand, C. Marlin, F. Montreuil, E. Nogues, M. Pelcat, Electro-Magnetic Side-Channel Attack Through Learned Denoising and Classification, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 2882–2886. doi:10.1109/ICASSP40776.2020.9053913.

[11] S. Fernández, E. Martínez, J. Varela, P. Musé, F. Larroca, Deep-TEMPEST: Using Deep Learning to Eavesdrop on HDMI from its Unintended Electromagnetic Emanations, in: Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing (LADC '24), 2024. doi:10.1145/3697090.3697094.

[12] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising, IEEE Transactions on Image Processing 26 (2017) 3142–3155. doi:10.1109/TIP.2017.2662206.

[13] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, R. Timofte, Plug-and-play image restoration with deep denoiser prior, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2022) 6360–6376. doi:10.1109/TPAMI.2021.3088914.

A. Online Resources

The sources for our modified TempestSDR, dataset, and trained models are available online:

- Modified TempestSDR on GitHub:
  https://github.com/filippt1/TempestSDR_Enhanced,
- Dataset on Hugging Face:
  https://huggingface.co/datasets/filippt1/TempestSDR_Enhanced_Dataset,
- Trained models on Google Drive:
  https://drive.google.com/drive/folders/1zFWvRVtZ-9s4WG3DEF2Kivu6meQ9UQvB?usp=sharing.