MOSAIC-SR: AN ADAPTIVE MULTI-STEP SUPER-RESOLUTION METHOD FOR LOW-RESOLUTION 2D BARCODES

Enrico Vezzali[⋆,†]*, Lorenzo Vorabbi*[†]*, Costantino Grana*[⋆]*, Federico Bolelli*[⋆] ✉

[⋆]University of Modena and Reggio Emilia, Italy
[†]Datalogic, S.p.A, Bologna, Italy

ABSTRACT

QR and Datamatrix codes are widely used in warehouse logistics and high-speed production pipelines. Still, distant or small barcodes often yield low-pixel-density images that are hard to read. Conventional solutions rely on costly hardware or enhanced lighting, raising expenses and potentially reducing depth of field. We propose *Mosaic-SR*, a multi-step, adaptive super-resolution (SR) method that devotes more computation to barcode regions than uniform backgrounds. For each patch, it predicts an uncertainty value to decide how many refinement steps are required. Our experiments show that Mosaic-SR surpasses state-of-the-art SR models on 2D barcode images, achieving higher PSNR and decoding rates in less time. All code and trained models are publicly available at `https://github.com/Henvezz95/mosaic-sr`

***Index Terms*—** Barcodes, QR Codes, Super Resolution

1. INTRODUCTION

Two-dimensional barcodes, such as QR codes and Datamatrix, are increasingly employed in warehouse inventory management systems [1] and component tracking in production lines [2]. In *warehouses*, workers often scan packages stacked on high shelves to track incoming and outgoing stock. Achieving a longer scanning range would save time and reduce the effort of manually retrieving items for closer scanning. In *component tracking pipelines*, parts rush along conveyor belts, each labeled with barcodes to identify their location and status in real-time. However, capturing a clear barcode under high-speed movement or from a significant distance can be challenging, leading to low-resolution or blurred images. Historically, improving such images relied on advanced camera hardware or stronger lighting, which raises costs and reduces the depth of field (DoF). *Super-resolution* (SR) provides a software-based alternative to enhance readability. In warehouses, SR can extend the functional scanning range. In pipelines, SR compensates for low-resolution sensors, which have larger pixels that collect more light, allowing shorter exposures and reducing motion blur. In many industrial applications, *speed* remains paramount: not only do some setups run on embedded hardware but also many require tens or hundreds of scans per second to ensure timely decoding. The advantages of 2D barcode SR would extend to other usages in retail and mobile applications.

Related Work in 2D Barcode SR. Early barcode-focused SR approaches include Kato *et al.* [3], who combined multiple low-res QR images via a binary constraint prior. More recently, Shindo [4] proposed a convolutional architecture (QR-CNN/QRGAN) for QR code enhancement.

Related Work in Real-Time SR. A broader body of literature addresses real-time SR for general images. FSRCNN [5] has been the first architecture targeting real-time performance, followed by ESPCN [6], which popularized the use of pixel-shuffle layers. IMDN [7] utilized cascaded blocks with an information distillation mechanism, while ECBSR [8] utilized over-parameterization [9] to enhance learning capabilities. Michelini and Lu [10], proposed three SR architectures for hardware-constrained devices (eSR-MAX, eSR-TM, eSR-TR). Finally, in 2023, RT4KSR [11] and AsConvSR [12] enabled real-time 4K upscaling. RT4KSR uses a separate architectural branch for high-frequency enhancement alongside over-parameterization during training. AsConvSR leverages assembled convolutions which can adapt convolution kernels according to the input features.

Our Contributions. Unlike most SR tasks, barcode images are typically grayscale with high-frequency edges and corners, and weak correlations between nearby regions—posing unique challenges to general-purpose SR models. We propose *Mosaic-SR*, a multi-step, adaptive super-resolution framework dedicated to 2D barcodes. It selectively devotes more computational time to likely barcode regions than uniform backgrounds, deciding how many refinement steps each patch requires based on an uncertainty estimate. Mosaic-SR outperforms existing SR methods on barcode images, yielding higher PSNR and decoding rates in less time. We provide all source code, and trained models, and rely on open-source libraries to ensure reproducible results.

2. PROPOSED METHOD

Multi-Step Super-Resolution. We address the problem of $2\times$ super-resolution, where each pixel $I(x, y)$ in a low-

✉ Corresponding author: *federico.bolelli@unimore.it*

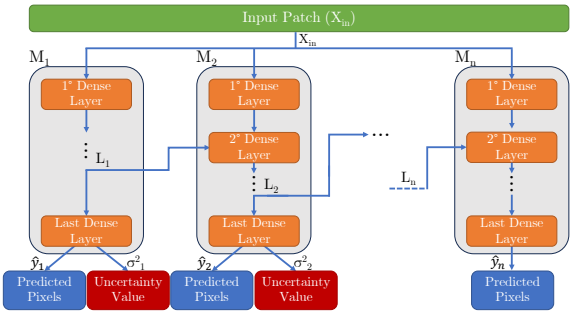


Fig. 1. Our proposed architecture for Multi-step super resolution. Each module is a fully-connected network that upscales a patch and generates an uncertainty value on the prediction.

resolution image I corresponds to a 2×2-pixel block in the high-resolution image I_{HD}. To formalize this, we seek a function f_θ that, given a local patch around $I(x, y)$, predicts the corresponding 2×2 block $N^{\text{HD}}_{2\times2}(x, y)$ in I_{HD}. Since a single pixel $I(x, y)$ alone provides insufficient information to reconstruct $N^{\text{HD}}_{2\times2}(x, y)$, we instead provide f_θ with $N_{k\times k}(x, y)$, a $k \times k$ patch centered at (x, y). For simplicity, we assume k is odd so that $I(x, y)$ lies at the center. This larger local context helps recover edges, corners, and other high-frequency details lost in the low-resolution input.

Low-resolution 2D barcodes (e.g., QR, Datamatrix) are demanding for SR due to sharp edges and corners, where small positional shifts can cause large intensity changes. Meanwhile, uniform backgrounds (e.g., paper) are simpler. We thus develop a multi-step architecture that concentrates more computation on complex, high-frequency areas while reducing effort on uniform regions. To achieve this, we propose a multi-step architecture in which each step refines the prediction of $N^{\text{HD}}_{2\times2}(x, y)$ given the local patch $N_{k\times k}(x, y)$. Every step yields a usable output and an uncertainty estimate. If that uncertainty is under a threshold, we stop; otherwise, we proceed. By adapting the number of refinement steps to local complexity, we ensure fidelity in key barcode regions while optimizing for speed elsewhere.

In Fig. 1, we illustrate the overall design. For each pixel $I(x, y)$, we extract $N_{k\times k}(x, y)$, flatten it into $X_{\text{in}}(x, y)$, and feed this into a fully connected network M_1. This model generates three outputs: (*i*) $\hat{y_1}$, a 4-element vector predicting the intensities of the pixels in $N^{\text{HD}}_{2\times2}(x, y)$; (*ii*) σ^2_1, a scalar uncertainty estimate, used to decide whether to refine further; (*iii*) L_1, a latent representation from the penultimate layer of M_1. If $\sigma^2_1 < Th_1$, $\hat{y_1}$ is the final prediction for $N^{\text{HD}}_{2\times2}(x, y)$. Otherwise, we invoke the next model, M_2. Each subsequent network M_i is a fully connected network that refines the previous prediction by taking as input $X_{\text{in}}(x, y)$, plus the previous latent representation L_{i-1}. Specifically, L_{i-1} is concatenated with the output of the first layer of M_i rather than appending it directly to X_{in}. This setup enables the first layer to extract new features from X_{in} more effectively and is usually faster because it limits the size of the input of the first layer. For n total steps, each intermediate M_i (where $i < n$) outputs a prediction $\hat{y}_i$ of $N^{\text{HD}}_{2\times2}(x, y)$, an uncertainty value σ^2_i and a latent representation L_i. Again, if $\sigma^2_i < Th_i$, we stop; otherwise, we continue. The final model M_n outputs only a refined prediction $\hat{y}_n$. Thresholds $\{Th_1, \ldots, Th_{n-1}\}$ are set before inference and affect image quality: higher thresholds yield faster inference but lower overall quality, while lower thresholds improve quality at the expense of more computation.

This architecture is loosely inspired by GrowNet [13], but differs in three main ways: (i) each step outputs a standalone prediction instead of combining intermediate results; (ii) each step's uncertainty estimate σ^2_i indicates when to stop; and (iii) latent representations are concatenated after the first layer, not directly with X_{in}.

Variance Prediction. So far, we have focused on predicting $N^{\text{HD}}_{2\times2}(x, y)$ from $N_{k\times k}(x, y)$. Next, we want to estimate the probability distribution of the prediction error and use it as a stopping criterion. Assuming a Gaussian error with zero mean, fully characterizing this distribution amounts to predicting its variance σ^2. In our proposed setup, each network M_i receives an input vector X_{in} (the flattened patch) and outputs $\hat{y}_i$ and $\sigma^2_i(X_{\text{in}})$. Here, $\hat{y}_i$ is the predicted mean of the output distribution (i.e., the super-resolved intensities), and $\sigma^2_i(X_{\text{in}})$ is the estimated error variance conditioned on X_{in}. Following the analysis of Nix and Weigend [14], M_i can be viewed as a maximum-likelihood estimator for $P(\hat{y}_i + e \mid X_{\text{in}})$, where e is the prediction error. If e is normally distributed with mean zero, minimizing its negative log-likelihood reduces to optimizing the cost C, defined as:

$$C = \frac{(\hat{y}_i - y_{\text{true}})^2}{\sigma^2_i(X_{\text{in}})} + \ln\big(\sigma^2_i(X_{\text{in}})\big), \tag{1}$$

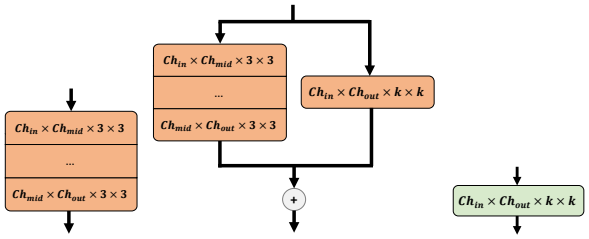
where y_{true} is the ground truth vector obtained by flattening $N^{\text{HD}}_{2\times2}(x, y)$. Using C alone can cause large initial variances to dominate training, hindering the ability to learn. Nix and Weigend [14] address this by training the mean first (via MSE loss) and fine-tuning with C. We instead propose doing both simultaneously via a combined loss:

$$\mathcal{L}_i = ((\hat{y}_i - y_{\text{true}})^2 + \alpha\, C), \tag{2}$$

where α is a small scalar. If $\alpha \ll 1$, the network is initially driven to minimize $(\hat{y}_i - y_{\text{true}})^2$, ensuring good mean predictions before shifting emphasis toward variance estimation.

Training Strategy. During training, each patch is processed by all n steps (i.e., $M_1, M_2, \ldots, M_n$). Notably, each model M_i can be reinterpreted as a convolutional network (CNN_i), where the first layer corresponds to an $k \times k$ kernel and subsequent layers to 1×1 convolutions. The weights will be the same but reshaped. This reshaping facilitates efficient GPU-based training. The overall training loss, for each patch, is as follows:

$$\mathcal{L}_{\text{total}} = \sum_{i=1}^{n} \mathcal{L}_i, \tag{3}$$



(a) ExpandNets (b) Ours (c) Inference

Fig. 2. Over-parameterization strategies during training: (a) ExpandNets, (b) Our Proposal. In (c) we see how architectures (a) and (b) are converted in the inference model.

where $\mathcal{L}_i$ is the combined loss, Eq. (2), for network M_i. Finally, we take the mean of each patch loss. However, optimizing all M_i jointly can lead to conflicting gradients. Instead, we adopt a sequential approach:

1. Train M_1 alone, learning $\hat{y}_1$ and σ_1^2 until convergence;
2. Freeze M_1 and train M_2 to refine predictions from X_{in} and L_1;
3. Repeat for $M_3, \ldots, M_n$, each time freezing previous models;
4. Finally, unfreeze all weights and train the architecture end-to-end.

This procedure prevents instability from competing updates, yet allows the entire network to benefit from end-to-end fine-tuning at the last stage.

Model Over-Parameterization. To improve training stability and performance, we employ *model over-parameterization*, as proposed in recent works [9, 15, 16]. This technique trains networks with additional linear layers (stacked or parallel), which collapse into a single equivalent layer at inference time. It ensures faster, more stable training while maintaining a compact model for deployment, a strategy widely adopted in real-time super-resolution [8, 11, 17]. In each network M_i, we first targeted the large $k \times k$ kernel that extracts crucial information from the input patch. Inspired by ExpandNets [16], we replaced a single $k \times k$ layer with a cascade of $\frac{k-1}{2}$ layers with 3×3 kernels. We then modified the ExpandNets approach by adding a parallel $k \times k$ branch to the 3×3 cascade for better feature extraction, as shown in Fig. 2. Based on positive results with kernel expansion, we also applied the *expanding convolutional layer* strategy from ExpandNets [16] to all remaining layers. Each 1×1 layer was replaced by three consecutive layers with kernel 1×1 and an expansion factor of 2. This further stabilizes training and improves performance, yet still collapses to a single layer for inference.

Inference. After training, we apply two structural re-parameterization steps. First, the over-parameterized linear blocks become single convolutional layers. Next, the resulting CNNs are converted to fully connected models, as inference operates on flattened patches. Fig. 3 illustrates our inference pipeline: we transform the input image into a batch of flattened patches via an optimized `im2col` function, process them with $\{M_1, \ldots, M_n\}$, and then use a `col2im` function to reassemble the final upscaled image. During inference, the pre-defined thresholds $\{Th_1, \ldots, Th_{n-1}\}$ guide selective refinement: all patches go to M_1, and for each patch, if $\sigma_i^2 \geq Th_i$, it proceeds to M_{i+1} for further enhancement.

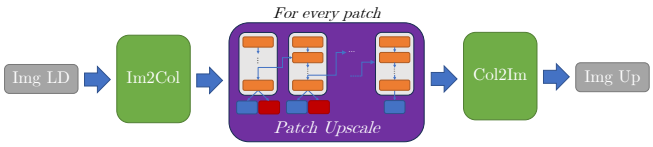


Fig. 3. Super resolution pipeline at inference time.

3. EXPERIMENTAL RESULTS

3.1. Experimental Setup

Dataset. In our tests, we used the publicly available BarBeR[1] dataset [18, 19, 20], which contains 8 748 barcode images, including 1 756 2D barcodes. From these, we selected only QR, Datamatrix, and Aztec crops with a minimum density of 3.2 PPM, producing 1 366 valid crops. Each crop was resized via bicubic interpolation to yield seven low-resolution (LR) variants (1.0 PPM to 1.6 PPM, with a step of 0.1 PPM) and seven corresponding high-resolution (HR) versions at double each LR density (2.0 PPM to 3.2 PPM). This process generated 9 562 LR/HR pairs. All images are 128×128 pixels, converted to grayscale, and contrast stretched to [0,1]. When we decrease the pixel density, the background proportion increases accordingly. We performed *4-fold cross-validation* to ensure a higher reliability of the results.

Hardware Setup. The primary goal of our method is to run efficiently on CPUs, aligning with real-world barcode applications. To evaluate performance, we benchmarked on two contrasting systems: a high-end PC (AMD Ryzen Threadripper Pro 5965WX) and a Raspberry Pi 3B+ (1.4 GHz quad-core ARM CPU).

Code Implementation. All training and testing routines are written in Python. During tests, each model is converted to TensorFlow Lite for optimization. To speed up the image conversion to patches, we implemented a SIMD-optimized `im2col` kernel in C++, reducing runtime to a mean of 75μs on the PC. Then, we integrated it using a Ctypes interface. Additional preprocessing steps were accelerated with Numba, ensuring an efficient overall pipeline.

3.2. Comparison with State-of-the-arts

Model Parameters. We train three upscaling networks $\{M_1, M_2, M_3\}$ with a 7×7 input patch. M_1 has 3 layers and

[1] `https://ditto.ing.unimore.it/barber/`

Table 1. PSNR, SSIM, and decoded barcode count for baseline models compared to our method, with processing times on PC and Raspberry Pi.

Method	PSNR (dB, ↑)	SSIM (↑)	# Decodings (↑)	Time PC (ms, ↓)	Time Raspberry (ms, ↓)
Bicubic (openCV)	15.35	0.705	2 336	0.046	1.13
eSR-MAX K5C8 [10]	15.62	0.772	2 657	0.887	26.60
Ours (0.25^2, 0.25^2)	**15.77**	**0.782**	**3 330**	**0.883**	**23.29**
ECBSR M4C8 [8]	15.81	**0.788**	3 502	2.195	62.83
RT4KSR XXS [11]	15.64	0.775	2 900	2.182	55.73
RT4KSR S [11]	15.64	0.776	2 901	3.413	86.70
Ours (0.13^2, 0.13^2)	**15.83**	0.785	**3 517**	**1.688**	**44.29**
AsConvSR [12]	15.59	0.772	2 815	8.778	200.2
ESPCN [6]	15.84	0.784	3 596	7.121	198.2
eSR-TM K7C16 [10]	15.65	0.777	2 940	9.458	238.7
eSR-TR K7C16 [10]	15.74	0.781	3 208	9.938	318.6
FSRCNN [5]	15.85	0.787	3 602	13.45	442.9
IMDN RTC [7]	15.83	**0.790**	3 695	10.97	356.6
RT4KSR XL [11]	15.64	0.776	2 893	6.586	178.9
QRCNN [4]	15.77	0.780	3 488	172.1	3 162
Ours (0.05^2, 0.07^2)	**15.90**	0.788	**3 714**	**4.221**	**109.9**

1 154 parameters, M_2 has 4 layers and 3 778 parameters, and M_3 has 4 layers and 14 274 parameters. We use Leaky ReLU ($\alpha = 0.1$) for each hidden layer, while the output layer uses a sigmoid activation. Each network is trained for 100 epochs (batch size of 16, Adam optimizer, learning rate 1×10^{-3}), halving the learning rate whenever the training loss stagnates for 15 epochs. Then, we unfreeze all weights and train further with a learning rate of 1×10^{-4}, again halving under the same stagnation condition until it falls below 1×10^{-6}.

Reference Models. We compare our model against representative state-of-the-art models in real-time super-resolution: AsConvSR [12], RT4KSR (XXS, S, XL) [11], eSR-MAX ($k = 5$, $c = 8$), eSR-TM ($k = 7$, $c = 16$), eSR-TR ($k = 7$, $c = 16$) [10], ECBSR ($M = 4$, $C = 8$) [8], IMDN RTC [7], ESPCN [6] and FSRCNN [5]. We also included QRCNN [4], specifically proposed for QR code SR. All models were trained on the BarBeR dataset following original strategies but using a single input channel.

Comparison Results. We measure the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and the number of successfully decoded barcodes by pyzbar, a Python wrapper of zbar. For each image, the detection is run five times and the lowest timing value is taken. This is done to remove the effect of external factors. All measures are taken using a single thread. PSNR and SSIM are calculated only on the pixels of the 2D barcodes since this is the main focus. Fig. 4 shows that, for a given mean processing time, our method decodes more barcodes than any other approach when appropriate thresholds are selected. We tested multiple combinations of thresholds Th_1 and Th_2, selecting values between 0 and 0.25^2 under the condition $Th_1 \leq Th_2$. Thresholds are expressed as squares to better represent them in terms of standard deviation. In Tab. 1, we group baselines by their processing times on a PC: (i) below 1 ms, (ii) between 1 ms and 4 ms, and (iii) above 4 ms. We select appropriate thresholds (Th_1 and Th_2) for our method to satisfy each time constraint. Across all groups, our method is the fastest (on both PC and Raspberry Pi), ensures the highest number of decodings, and achieves the highest PSNR. For instance, we decode 42.5 % more than the nearest competitor with slightly lower latency. In the mid-speed group, it matches the top decoding rate (ECBSR) but uses only 76.9 % of the time on PC and 70.5 % on Raspberry Pi. In the slowest group, it exceeds the best competitor (IMDN RTC) by a small margin in decoding, finishing in 38.5 % of the time on PC and 30.8 % of the time on Raspberry Pi. Our consistently high PSNR reflects how devoting extra computation to challenging regions effectively reduces outliers. Conversely, our method is not the highest scoring in terms of SSIM. We believe this is due to our loss function and the interleaving of outputs from different models, which may reduce smoothness but does not seem to hurt decoding accuracy (Tab. 1). Fig. 5 compares visual results, while Fig. 6 shows decoding rates across various pixels-per-module (PPM) values. We compare our method using the previously selected thresholds against the next-best method

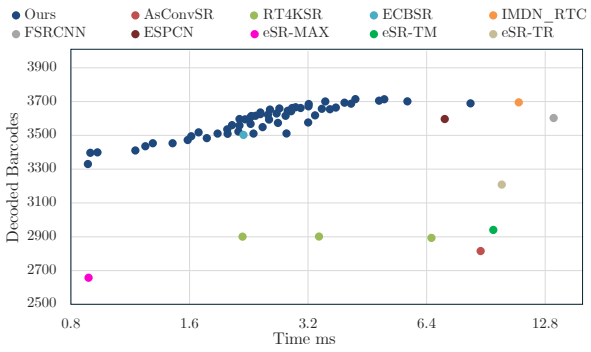


Fig. 4. Number of decoded barcodes by different types of upscaling methods. The x-axis shows the computational time on the test PC.

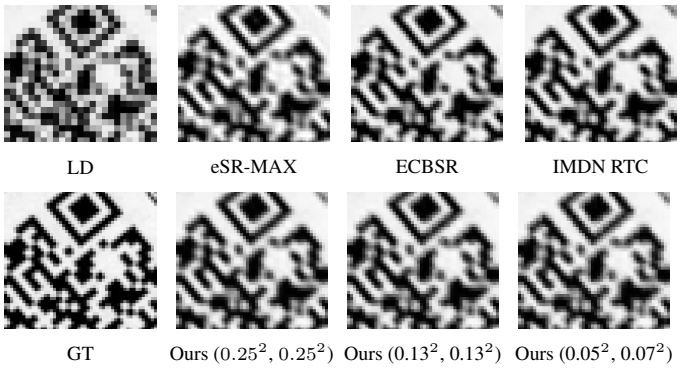


Fig. 5. Comparison of different upscaling methods on a crop of a 1.3PPM QR Code. The numbers in parenthesis indicate the values of Th_1 and Th_2.

Table 2. Results in terms of PSNR, SSIM and number of decodings of the models M_1, M_2 and M_3 with and without applying over-parameterization (op) during training.

Upscaled output	PSNR (dB, ↑)	SSIM (↑)	# Decodings (↑)
M_1 - wth op	15.77	0.782	3 330
M_1 - w/o op	15.68	0.777	3 212
M_2 - wth op	15.85	0.786	3 511
M_2 - w/o op	15.78	0.782	3 360
M_3 - wth op	15.91	0.788	3 689
M_3 - w/o op	15.85	0.786	3 587

for each time group. For reference, we also include bicubic upscaling. Without upscaling, none of the LR patches are decoded, showing the necessity of super-resolution for this task.

Efficacy of Variance Prediction. We evaluated whether the model accurately predicts error variance at each step by extracting 2.24×10^7 patches from the dataset. For each patch, we calculated the actual error and compared it to the predicted variance from M_1 and M_2. For each patch, we used models that had never seen the patch during training. The predicted standard deviation (square root of variance) was binned into 100 intervals between 0 and 0.25, and the root mean squared error of the patches in each bin was recorded. Ideally, these values should match the bin's central predicted standard deviation if the model is well-calibrated. As shown in Fig. 7, the empirical curve closely follows the ideal line up to 0.15. Beyond that, predictions diverge slightly but remain monotonic. This is sufficient for our method to work since it still ranks patches by error standard deviation.

3.3. Ablation Studies

Utility of Model Over-Parameterization. We trained three models identical to $\{M_1, M_2, M_3\}$ but without over-parameterization during training. Tab. 2 shows that over-parameterization consistently improves PSNR and SSIM while increasing the number of decoded barcodes for each M_i.

Input Patch Size. The input patch size plays a crucial role: larger patches provide more context but increase computational cost. In our experiments, 7×7 delivered the best results in terms of speed and performance. Fig. 8 compares 5×5 and 9×9 against 7×7, using multiple thresholds. While 5×5 can be slightly faster under certain thresholds, it significantly reduces image quality. Surprisingly, 9×9 is slower and less accurate, likely due to additional context not translating into useful features, which destabilizes training.

4. CONCLUSIONS

In this paper, we introduced Mosaic-SR, a multi-step, adaptive super-resolution algorithm tailored for 2D barcodes. Mosaic-SR devotes more computational effort to patches likely containing barcodes than to uniform backgrounds. It achieves this by estimating each patch's error variance and refining high-variance patches through additional steps. Since barcodes feature numerous edges and corners, they exhibit higher variance and benefit most from extra refinements. Our experiments demonstrate that Mosaic-SR surpasses state-of-the-art SR methods on barcode images, achieving higher PSNR and decoding rates in less time. While our focus was on 2D barcodes, Mosaic-SR's principles could extend to applications like enhancing text in scanned documents, refining satellite and medical images, and highlighting small manufacturing defects. These examples, which could involve high-frequency regions and uniform backgrounds, align well with Mosaic-SR's adaptive strategy. Finally, we have open-sourced our code and trained models, ensuring reproducibility and paving the way for future research.

Acknowledgements. Funding from UNIMORE and Fond. Mo. is received through the FAR 2024 (E93C24002080007).

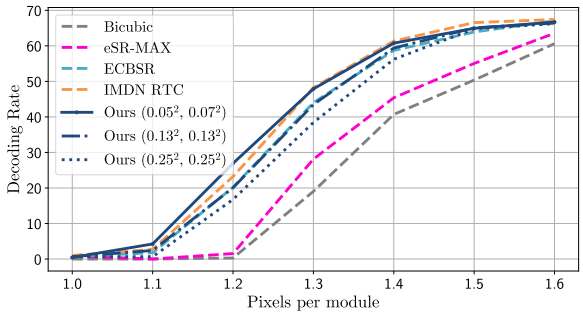


Fig. 6. Decoding rate depending on the PPM.

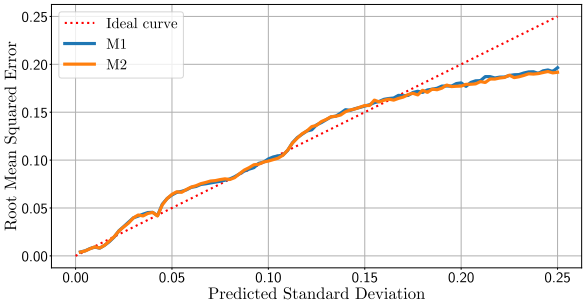


Fig. 7. Change of the root mean squared error at the change of predicted standard deviation.

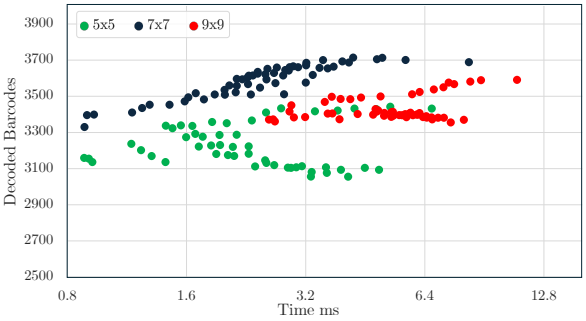


Fig. 8. Decodings *vs* time depending on the input patch size.

5. REFERENCES

[1] Shayantani Kar, Shresth Bhimrajka, Aditya Kumar, and Subhamita Mukherjee, "Mobile based Inventory Management System with QR code," in *2022 IEEE International Conference on Electronics, Computing and Communication Technologies*, 2022.

[2] Fei Chen, Yuxi Luo, Nektarios Georgios Tsoutsos, Michail Maniatakos, Khaled Shahin, and Nikhil Gupta, "Embedding Tracking Codes in Additive Manufactured Parts for Product Authentication," *Advanced Engineering Materials*, vol. 21, no. 4, 2019.

[3] Yuji Kato, Daisuke Deguchi, Tomokazu Takahashi, Ichiro Ide, and Hiroshi Murase, "Low Resolution QR-Code Recognition by Applying Super-Resolution Using the Property of QR-Codes," in *2011 International Conference on Document Analysis and Recognition*, 2011.

[4] Takahiro Shindo, Taiju Watanabe, Remina Yano, Marika Arimoto, Miho Takahashi, and Hiroshi Watanabe, "Super Resolution for QR Code Images," in *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE)*, 2022.

[5] Chao Dong, Chen Change Loy, and Xiaoou Tang, "Accelerating the Super-Resolution Convolutional Neural Network," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, 2016.

[6] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[7] Jin Wang, Yiming Wu, Shiming He, Pradip Kumar Sharma, Xiaofeng Yu, Osama Alfarraj, and Amr Tolba, "Lightweight Single Image Super-Resolution Convolution Neural Network in Portable Device," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 11, 2021.

[8] Xindong Zhang, Hui Zeng, and Lei Zhang, "Edge-oriented Convolution Block for Real-time Super Resolution on Mobile Devices," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021.

[9] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun, "RepVGG: Making VGG-style ConvNets Great Again," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[10] Pablo Navarrete Michelini, Yunhua Lu, and Xingqun Jiang, "edge-SR: Super-Resolution For The Masses," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

[11] Eduard Zamfir, Marcos V Conde, and Radu Timofte, "Towards Real-Time 4K Image Super-Resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[12] Jiaming Guo, Xueyi Zou, Yuyi Chen, Yi Liu, Jia Hao, Jianzhuang Liu, and Youliang Yan, "AsConvSR: Fast and Lightweight Super-Resolution Network with Assembled Convolutions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[13] Sarkhan Badirli, Xuanqing Liu, Zhengming Xing, Avradeep Bhowmik, Khoa Doan, and Sathiya S Keerthi, "Gradient Boosting Neural Networks: GrowNet," *arXiv preprint arXiv:2002.07971*, 2020.

[14] David A Nix and Andreas S Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, 1994, vol. 1.

[15] Sanjeev Arora, Nadav Cohen, and Elad Hazan, "On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization," in *International Conference on Machine Learning*, 2018.

[16] Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann, "ExpandNets: Linear Over-parameterization to Train Compact Convolutional Networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[17] Liu Ding Du Zongcai, Liu Jie, Tang Jie, Wu Gangshan, and Fu Lean, "Fast and Memory-Efficient Network Towards Efficient Image Super-Resolution," in *NTIRE (CVPR Workshop)*, 2022.

[18] Enrico Vezzali, Federico Bolelli, Stefano Santi, and Costantino Grana, "Barber: A barcode benchmarking repository," in *International Conference on Pattern Recognition*. Springer, 2025, pp. 187–203.

[19] Enrico Vezzali, Federico Bolelli, Stefano Santi, Costantino Grana, et al., "BarBeR-Barcode Benchmark Repository: Implementation and Reproducibility Notes," in *Proceedings of 5th International Workshop on Reproducible Research in Pattern Recognition, RRPR 2024*, 2025.

[20] Enrico Vezzali, Federico Bolelli, Stefano Santi, and Costantino Grana, "State-of-the-art Review and Benchmarking of Barcode Localization Methods," *Engineering Applications of Artificial Intelligence*, pp. 1–29, 2025.