Latent Neural Cellular Automata for Resource-Efficient Image Restoration

Andrea Menta[1], Alberto Archetti[1], and Matteo Matteucci[1]

[1]Politecnico di Milano, Milan, Italy
name.surname@polimi.it

Abstract

Neural cellular automata represent an evolution of the traditional cellular automata model, enhanced by the integration of a deep learning-based transition function. This shift from a manual to a data-driven approach significantly increases the adaptability of these models, enabling their application in diverse domains, including content generation and artificial life. However, their widespread application has been hampered by significant computational requirements. In this work, we introduce the Latent Neural Cellular Automata (LNCA) model, a novel architecture designed to address the resource limitations of neural cellular automata. Our approach shifts the computation from the conventional input space to a specially designed latent space, relying on a pre-trained autoencoder. We apply our model in the context of image restoration, which aims to reconstruct high-quality images from their degraded versions. This modification not only reduces the model's resource consumption but also maintains a flexible framework suitable for various applications. Our model achieves a significant reduction in computational requirements while maintaining high reconstruction fidelity. This increase in efficiency allows for inputs up to 16 times larger than current state-of-the-art neural cellular automata models, using the same resources.

Introduction

Cellular Automata (CA) (von Neumann, 1951) are a computational model for simulating the dynamics of complex systems from simple local interactions. CAs operate on a regular lattice in which each cell is in one of a finite number of states. At each discrete time step, cells are updated using simple local rules based on the state of neighboring cells. CAs are capable of generating complex patterns from simple update rules. The most famous example is John Conway's Game of Life, which is Turing complete (Rendell, 2002). In general, the potential to generate intricate global patterns from simple local rules has always attracted interest in CA research.

The advent of Deep Learning (DL) (Goodfellow et al., 2016) has revolutionized various domains, such as computer vision and natural language processing, establishing itself as a State-of-the-Art (SotA) methodology. Its inherent flexibility and generalization power have allowed several extensions of existing models, including CAs. Neural Cellular Automata (NCA) (Mordvintsev et al., 2020) exemplify this progression, by integrating DL into CAs. NCAs differ from CAs by using neural networks as their transition functions. This aspect allows the transition from manually designed rules to data-driven learning processes that target specific behaviors. Although many promising applications have been proposed, ranging from content generation (Sudhakaran et al., 2021) to control (Variengien et al., 2021), NCAs face significant limitations due to their resource requirements. The need for numerous concurrent updates introduces significant latency and memory overhead (Tesfaldet et al., 2022), similar to Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986), hindering their practical application.

To address these challenges, we introduce the Latent Neural Cellular Automata (LNCA) model. The core of LNCA is to move computations from the conventional input space to a customized, compressed latent space. The idea is to distill the essential information for the NCA to solve the problem into a lower-dimensional manifold, leaving the accessory information to a secondary component. For this purpose, an Autoencoder (AE) is used to compress the input and learn an embedding function. Our research focuses primarily on Image Restoration (IR), tackling denoising and deblurring tasks. We evaluate LNCA against SotA solutions such as Restormer (Zamir et al., 2022) and NAFNet (Chen et al., 2022), which are based on standard deep learning techniques, as well as the best NCA model, called ViTCA (Tesfaldet et al., 2022). Our evaluation includes 10 datasets, both real and synthetic, measuring reconstruction performance and computational efficiency.

To the best of our knowledge, LNCA is the first model applying NCAs in the latent space of a DL autoencoder backbone. Thus, the contributions of this work are twofold:

- We propose a novel architecture called LNCA and the corresponding learning process to mitigate the inherent limitations of NCA models by exploiting the expressiveness of DL autoencoders.

- We apply the LNCA architecture to IR tasks, achieving significant reductions in latency and GPU memory usage

while maintaining competitive reconstruction performance compared to SotA models.

To facilitate further research, we have made the source code of the experiments publicly available on GitHub.

Related Work

This section describes the main works related to (neural) cellular automata and image restoration.

Cellular Automata

Cellular Automata (CA) are defined as a tuple $\mathcal{M} = (\mathcal{L}, \mathcal{S}, \eta, \phi)$, where $\mathcal{L}$ is a d-dimensional lattice, $\mathcal{S}$ is the set of possible states, $\eta : \mathcal{L} \to 2^{\mathcal{L}}$ is the neighborhood function and $\phi : \mathcal{S}^n \to \mathcal{S}$ is the transition function leveraging an n-sized neighborhood. Let $s_i^t \in \mathcal{S}$ indicate the state of cell i at time step t and $\Omega_i^t = \left\{ s_j^t : j \in \eta(i) \right\}$ the set of neighbors of cell i according to η. Then, the state update for cell i at each iteration t is computed as

$$s_i^{t+1} = \phi\left(\left\{ s_i^t \cup \Omega_i^t \right\}\right).$$

Neural Cellular Automata (NCAs) extend the CA framework by replacing the transition function ϕ with a learnable neural network ϕ_θ with parameters θ, preserving the essential feature of locality. However, training the neural network ϕ_θ requires some modifications to standard CAs. First, the state representations are continuous. Second, optimization is performed after n processing iterations, and a boolean mask is used during each iteration to allow asynchronous updates, which improves training dynamics, as in (Mordvintsev et al., 2020). This training methodology typically involves a two-step process with a replay buffer to perform pool sampling. During each odd step, the neural network updates a newly initialized tensor x, and the resulting tensor x' is stored in the replay buffer. For even steps, instead, the network retrieves x' from the replay buffer, updates it, and then stores the new tensor x'' back in the buffer. This strategy ensures that the target output acts as an attractor in the training process, effectively preventing the neural network from deviating from the desired behavior.

Image Restoration

Image Restoration (IR) is a central task in digital image processing that aims to reconstruct corrupted images into their original form. This process involves reducing distortions such as noise and blur. Over time, a plethora of methods has been developed to tackle IR, ranging from basic filtering techniques (Banham and Katsaggelos, 1997) to advanced deep learning models, each improving the quality of the restoration. In our analysis, we focus on three state-of-the-art deep learning-based architectures, called Restormer, NAFNet, and ViTCA.

Restormer (Zamir et al., 2022), leverages a residual learning approach based on the classic UNet (Ronneberger et al., 2015) architecture, powered by Transformer blocks (Vaswani et al., 2017). Each block is composed of a Multi-Dconv Head Transposed Attention (MDTA) and a Gated-Dconv Feed-forward Network (GDFN). MDTA offers a memory-efficient alternative to the self-attention block by utilizing channel dimension, while GDFN incorporates a gating mechanism to the classic transformer head.

NAFNet (Chen et al., 2022) is a non-linear activation-free network, based on a UNet structure. The basic block of this architecture includes two components, the Simple Gate (SG) and the Simplified Channel Attention (SCA). SG introduces non-linearities in the block, while SCA performs an attention operation leveraging a channel-wise product. Both SG and SCA derive from the gated linear unit (Dauphin et al., 2017).

Finally, ViTCA (Tesfaldet et al., 2022) is the first application of the attention mechanism to the NCA family. This CA-based architecture implements a transition function composed of four stages: embedding, Multi-Head Self-Attention (MHSA), Multi-Layer Perceptron (MLP), and head. The embedding takes the seeded input (optionally including the positional encoding) and feeds it through a linear projection. The MHSA, containing the major novelty of ViTCA, estimates the relative importance of each token. It implements a localized self-attention method to respect NCA locality constraints, limiting the attention span to each cell's neighborhood. This way, even though information propagates locally, it achieves a global behavior. The MLP, composed of two fully connected layers with GELU activation (Hendrycks and Gimpel, 2023), completes the classic Vision Transformer architecture (Dosovitskiy et al., 2021), while the head reshapes the output obtaining the final update vector. Lastly, this model is trained using pool sampling and curriculum learning, a typical approach for NCA architectures.

Latent Neural Cellular Automata

Compared to classic Feed-forward Neural Networks, NCAs require multiple processing steps to obtain the desired output, penalizing latency and memory. While the former limitation is self-evident, the latter arises from the necessity to unroll the network (as in RNNs) to perform backpropagation during training. This paper introduces a novel approach to address these resource constraints, called Latent Neural Cellular Automaton (LNCA). The core concept of our methodology is to move the NCA computation – local information passing and state transition – from the input space to a specially designed latent space. The main goal motivating this shift is to limit the dimensionality of the space in which the NCA operates, extracting only the task-specific information required and reducing redundant operations. In the context of IR, our approach is particularly advantageous. We aim to build an embedding space featuring a distinct separation between the image semantics and the information about its corruption. This way, the autoencoder focuses on constructing an embedding space structured to decouple image and corrup-

tion information, while the NCA performs the restoration task. This task division significantly enhances the overall efficiency of the system. In the following, we present the LNCA architecture and its training pipeline in the IR task.

Architecture

The LNCA architecture is composed of two modules, the Autoencoder (AE) and the Neural Cellular Automaton (NCA). The AE's encoder module projects input images from the input space to the latent space, while the AE's decoder performs the inverse transformation. The NCA, instead, performs the actual image restoration operating in the latent space. Figure 1 shows the LNCA architecture. Specifically, the input image $x \in \mathbb{R}^{H \times W \times C}$ is translated by the encoder $\mathcal{E}$ of the AE into its latent representation $\hat{x} \in \mathbb{R}^{\hat{H} \times \hat{W} \times \hat{C}}$. Then, the NCA $\mathcal{N}$ performs n transition steps starting from the initial state $\hat{x}$ and obtaining a new latent representation $\hat{x}'$ with the same dimensionality. Finally, the decoder block $\mathcal{D}$ of the AE transforms the latent denoised image $\hat{x}'$ to the original image space, obtaining the desired output $x' \in \mathbb{R}^{H \times W \times C}$.

Autoencoder The AE's core objective is to create a lower dimensional manifold of the input space containing task-related information for the NCA while preserving reconstruction fidelity. For this reason, our architecture opts for a classical AE model instead of more sophisticated variants, such as variational autoencoders (Kingma and Welling, 2014), which may introduce additional reconstruction uncertainty. As shown in Figure 1, we adopted a standard AE architecture (Ronneberger et al., 2015) with some minor tweaks to obtain better convergence during training. We included a variant of the convolutional block, called adjusted convolution. This block comprises an initial convolution with 3×3 kernel and no activation, followed by batch normalization, and swish activation function. The only exception is the output layer which uses a sigmoid activation. The same is applied to the transposed version of convolution used in the decoder. This block generally provides superior convergence and stability properties for AE compared to standard convolutions.

To help the latent space construction, we made two major additions to the AE architecture, namely a skip-connection and an encoder side-channel. As skip-connections aid in reconstruction quality (i.e., passing less compressed information), they funnel image semantics unrelated to the corruption directly to the decoder. At the same time, all the restoration information flows into the AE's bottleneck for NCA processing. This behavior is enforced through a composite loss function, detailed in the Autoencoder Training Section.

Neural Cellular Automata The NCA is the component devoted to the actual restoration task. We opted to integrate a ViTCA-based NCA as the NCA core of our architecture, as it stands as the SotA within the models based on NCA. Additionally, this enables a direct comparison with the original, non-latent ViTCA and our novel model in terms of reconstruction performance and computational complexity.

To further reduce computational requirements and evaluate the impact of the attention mechanism for NCA architectures, we developed an alternative attention-less NCA model. This model, called NAFCA, is inspired by NAFNet, the architecture that replaced self-attention with a lighter gating mechanism, adjusted to satisfy the CA locality constraint. This is achieved by substituting the transformer-based transition function with a simpler architecture. Specifically, this model comprises three residual blocks: the perception block, the update block, and the head. The perception block performs the information extraction and it is composed of an initial layer normalization, followed by an activation-free 1×1 convolution and a Simple Gate (Chen et al., 2022). The classic attention mechanism is then substituted by $x' = x \circ \mathcal{C}(x)$, where $\mathcal{C}$ is a single-filter 3×3 convolution respecting the Moore neighborhood and $\circ$ is the Hadamard product. Then, another 1×1 convolution and a dropout layer complete the perception block. The update block takes care of the transition process. Its structure is similar to the perception block, except for the lack of attention mechanism and the number of filters in the convolutional layers. Finally, the head handles the state update. It is composed of a layer normalization and a 1×1 convolution.

Training

In the context of NCA, the training procedure must be carefully designed to allow model convergence. We chose a training strategy consisting of two phases to allow for more robust convergence. The first phase is the AE training, where the AE learns to build a latent space suitable for the NCA while minimizing the reconstruction error. The second phase is NCA training, where the NCA learns to reconstruct the image within the latent space. In each phase, the non-trained part of the architecture (i.e., first the NCA, then the AE) is kept frozen. This two-phase approach allows a clear separation of tasks between the AE and the NCA. Specifically, the AE manages faithful image reconstructions (with or without noise) from the embedding space and the NCA performs image restoration within the embedding space. Despite the apparent complexity of this approach, the training algorithm is designed to minimize the impact on training time.

Autoencoder Training The first training phase involves the AE block only. Thus, the NCA processing is bypassed with a direct connection between the encoder and the decoder. This phase leverages batches x composed of three elements:

- Anchor samples (x_A), the ground truth restored images.
- Positive samples (x_P), corrupted versions of x_A.
- Negative samples (x_N), derangement (permutation where no element appears in its original position) of x_A.

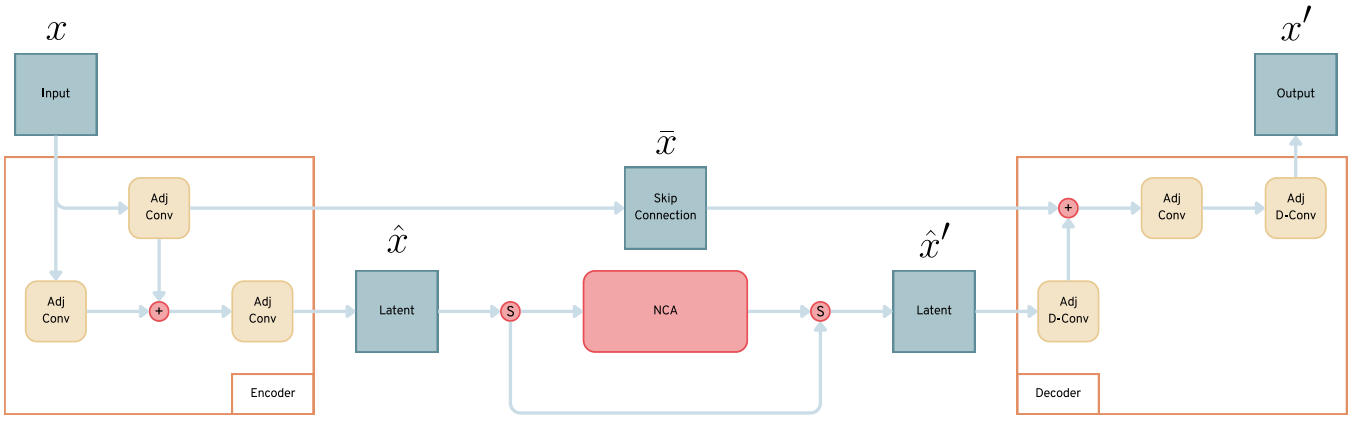

Figure 1: LNCA structure. $+$ is an element-wise addition. S is a channel switch. On the left, the Encoder, which is composed of adjusted convolution blocks, outputs the skip-connection $\bar{x}$ and the latent tensor $\hat{x}$. In the middle, the NCA block processes the latent tensor to remove the corruption, obtaining $\hat{x}'$. On the right, the Decoder uses both the skip-connection and the latent tensor to reconstruct the final output x'. The switch path around the NCA is used to bypass the NCA computation during the AE training, as described in the Autoencoder Training Section.

Given a sample batch, the AE training procedure is composed of two separate gradient descent steps with dedicated losses to ensure more stable convergence.

The first step includes the sum of three losses: the reconstruction loss, the distance loss, and the task loss. The goal of the reconstruction loss is to minimize the AE's reconstruction error and it is evaluated as the mean squared error between the AE's input and output tensor:

$$\mathcal{L}_{\text{REC}_{\text{AE}}}(x, x') = \text{MSE}(x, x').$$

The distance loss takes inspiration from the Triplet loss (Schroff et al., 2015) and its goal is to enforce the similarity in the latent space between images with the same subjects and the dissimilarity between images with different subjects. Given a tolerance margin α, this loss is calculated as

$$\mathcal{L}_{\text{DIST}}(\hat{x}) = \max\{0, \text{MSE}(\hat{x}_A, \hat{x}_P) - \text{MSE}(\hat{x}_A, \hat{x}_N) + \alpha\}.$$

Finally, the task loss is used to obtain a faithful reconstruction of the input corruption. To enforce this property, this loss takes into account only the corrupted pixels. Specifically, it applies a boolean mask before computing the mean squared error of the corrupted samples:

$$\begin{aligned}\mathcal{L}_{\text{TASK}}(x, x') &= \text{MSE}(x_P \circ m_{A,P}, x'_P \circ m_{A,P}) \\ m_{A,P} &= \mathbf{1}(x_A - x_P \neq 0).\end{aligned}$$

Each of these losses is then weighted and summed to obtain the loss of the first gradient descent step. The weights are determined through validation over a finite set of values using grid-search. This tuning procedure ensures a balance between reconstruction quality and latent space structure and helps manage differences in loss magnitudes.

The second gradient descent step, instead, involves the equivalence loss, inspired by the Swapping AE (Park et al., 2020) design. The goal of this loss is to split the information related to the image semantics and the image corruption between the two encoder outputs. In particular, the skip-connection $\bar{x}$ must help in the reconstruction of the image semantics, while the corruption profile is encoded only in the latent tensor $\hat{x}$. This is obtained by swapping $\hat{x}_P$ and $\hat{x}_A$, the latent tensors corresponding to the corrupted images and the ground truth images, respectively. In fact, by substituting $\hat{x}_P$ with its clean counterpart $\hat{x}_A$, while keeping the corrupted skip-connection $\bar{x}_P$, we should obtain a clean output x_A and vice versa. We enforce this property by minimizing the relative mean squared error:

$$\begin{aligned}\mathcal{L}_{\text{EQ}}(x, \bar{x}, \hat{x}) = &\text{MSE}(x_A \circ m_{A,P}, x^\star_A \circ m_{A,P}) + \\ &\text{MSE}(x_P \circ m_{A,P}, x^\star_P \circ m_{A,P}) \\ x^\star_A = &\mathcal{D}(\hat{x}_A + \epsilon, \bar{x}_P) \\ x^\star_P = &\mathcal{D}(\hat{x}_P, \bar{x}_A),\end{aligned}$$

where $\epsilon \sim N(0, v)$ is a smoothing factor that allows for a small tolerance in the reconstruction.

Neural Cellular Automata Training In this phase, we focus on the restoration process. Each input batch x is composed only of corrupted images and it is associated with a clean ground truth y. Furthermore, the generated latent tensor $\hat{x}$ is processed by the NCA module, obtaining $\hat{x}'$.

Similar to the AE training, this procedure leverages multiple losses to learn the IR task: the reconstruction loss, the latent loss, and the overflow loss. Each loss has an associated weight, which is tuned similarly to the AE case. First, the reconstruction loss minimizes the reconstruction error of the overall architecture in the image space. It is computed as the mean squared error between the predicted output x' and the

ground truth y:

$$\mathcal{L}_{\text{REC}_{\text{NCA}}}(y, x') = \text{MSE}(y, x').$$

The latent loss plays a similar role to $\mathcal{L}_{\text{REC}_{\text{NCA}}}$ but in the latent space. It enforces the similarity between the latent tensor of the ground truth, $\hat{y}$, and the one reconstructed by the NCA processing, $\hat{x}'$, starting from the associated corrupted input:

$$\mathcal{L}_{\text{LAT}}(\hat{y}, \hat{x}') = \text{MSE}(\hat{y}, \hat{x}').$$

Finally, the overflow loss regularizes the output magnitude of the NCA using an L1 norm. As in (Tesfaldet et al., 2022), it is applied to both the output and the hidden channels of the NCA output tensor $\hat{x}'$:

$$\mathcal{L}_{\text{OVER}}(\hat{x}') = \frac{1}{C}\left\|\hat{x}' - \text{clip}(\hat{x}')\right\|_1,$$

where C is the number of channels considered. Output channels are clipped in the interval $[0, 1]$, while hidden channels are in the interval $[-1, 1]$.

Experiments and Results

In this section, we evaluate the LNCA architecture in the IR task against several SotA models. In addition to the restoration performance, the analyses focus on time and memory resource efficiency, to assess the applicability of LNCA compared with existing NCA solutions.

Datasets

This section collects the information related to the datasets involved in our experiments. To allow for a comprehensive performance evaluation, we adopted both synthetic and real-world datasets. Concerning synthetic datasets, we started from CelebA (Liu et al., 2015), CIFAR-10 (Krizhevsky and Hinton, 2009), and TinyImageNet (Russakovsky et al., 2015) and applied specific corruption patterns. For denoising, we applied an additive Gaussian noise in a channel-coherent fashion. This procedure generates tone shifts rather than completely out-of-context pixel values, making the task harder. For deblurring, we use a camera motion blur. It generates a defocus along a random direction coherent for all the elements in the image.

For the real datasets, instead, we selected some of the most widely acknowledged in the image restoration field, namely Renoir (Anaya and Barbu, 2018) and SID (Chen et al., 2018) for denoising, and GoPro (Nah et al., 2017) and RealBlur (Rim et al., 2020) for deblurring.

Synthetic datasets grant fine-grained control over the type of corruption applied to the images, while real-world data allows for performance assessment in realistic scenarios. Also, synthetic and real datasets differ in their cardinality. On the one side, synthetic datasets contain many low-resolution images, while real datasets are composed of a few high-resolution images. Therefore, we need to apply different preprocessing to use these datasets properly. Specifically, for all the real ones, we applied a tile-shrink-clean procedure, which converts the original image into a $n \times m$ patchwork, resizes each tile to 32×32 pixels, and then removes redundant images. Conversely, for the synthetic datasets, we directly resized each image to the desired shape.

Training Procedure

In the following, we describe the experimental procedure adopted to collect the restoration performance and the time and memory requirements of LNCA and the baseline models from the literature. Specifically, we decided to use Restormer (Zamir et al., 2022) and NAFNet (Chen et al., 2022) as representatives of classic deep learning architectures, and ViTCA (Tesfaldet et al., 2022) as the NCA flagship model. On our side, as stated in the Neural Cellular Automata Section, we propose two versions of the LNCA architecture, with a different NCA structure. The first version, referred to as LatentViTCA, adopts ViTCA as the NCA module. Conversely, the second version, LatentNAFCA, has an attention-free NCA core, leading to a lighter architecture.

To keep the tests as fair as possible, we kept consistent training procedures across all the models. Specifically, we applied an 80-20% split to obtain training and test sets followed by another 80-20% split on the training set to extract the validation set. When training on synthetic datasets, we employ curriculum learning in the training set, feeding images of increasing corruption, while validation and test sets use a fixed difficulty. For real datasets, instead, we do not apply any curriculum learning.

Each model has been trained for 20 epochs, reaching a performance plateau. We opted for using Adam as an optimizer with cosine annealing scheduling (Loshchilov and Hutter, 2017) for the learning rate. Furthermore, regarding CA-based architectures, we used a pool size of 1024 and a hidden-channel size of 32, as in (Tesfaldet et al., 2022). During training, the number of CA iterations was uniformly sampled between 8 and 32, while for validation and testing, we fixed it to 64. This triplet of parameters is commonly used to account for pool sampling during training and ensure that the model achieves consistent performance.

Results on Restoration Performance

This section comments on the results obtained by LNCA and the SotA IR counterparts concerning pure restoration performance. The goal of our proposed latent models is to improve the computational and memory efficiency of NCA architectures at the expense of task utility. Therefore, we expect latent architectures to perform slightly worse than the alternatives in pure restoration performance.

To assess performance, we adopt the reference metric in IR, namely the Structural Similarity Index Measure (Wang et al., 2004) (SSIM). SSIM is a perception-based metric incorporating the concepts of luminance, contrast, and structure

Table 1: Denoising results on synthetic (CelebA, CIFAR-10, and TinyImageNet) and real (Renoir and SID) datasets, reporting the SSIM of each model. The best results are highlighted in bold, while the second best are underlined.

Models	CelebA	CIFAR-10	TinyImageNet	Renoir	SID
NAFNet	<u>0.958</u>	**0.924**	0.837	<u>0.962</u>	<u>0.752</u>
Restormer	**0.967**	<u>0.918</u>	**0.880**	**0.965**	**0.781**
ViTCA	0.942	0.910	<u>0.852</u>	0.939	0.647
LatentViTCA	0.889	0.855	0.802	0.877	0.646
LatentNAFCA	0.837	0.833	0.776	0.880	0.652

(a) Ground truth. (b) Noised.

(c) LatentViTCA. (d) LatentNAFCA.

(e) NAFNET. (f) Restormer. (g) ViTCA.

Figure 2: Denoising results over a $\mathbf{32} \times \mathbf{32}$ input sample.

Table 2: Deblurring results on synthetic (CelebA, CIFAR-10, and TinyImageNet) and real (GoPro and RealBlur) datasets, reporting the SSIM of each model. The best results are highlighted in bold, while the second best are underlined.

Models	CelebA	CIFAR-10	TinyImageNet	GoPro	RealBlur
NAFNet	<u>0.986</u>	<u>0.970</u>	<u>0.962</u>	<u>0.908</u>	<u>0.869</u>
Restormer	**0.993**	**0.977**	**0.975**	**0.934**	**0.918**
ViTCA	0.941	0.903	0.870	0.773	0.814
LatentViTCA	0.769	0.752	0.672	0.739	0.772
LatentNAFCA	0.745	0.739	0.673	0.739	0.771

(a) Ground truth. (b) Blurred.

(c) LatentViTCA. (d) LatentNAFCA.

(e) NAFNET. (f) Restormer. (g) ViTCA.

Figure 3: Deblurring results over a $\mathbf{32} \times \mathbf{32}$ input sample.

of an image, which is estimated as follows:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

where μ_x and μ_y are the sample means, σ_x and σ_y the standard deviations, σ_{xy} the cross-correlation, c_1 and c_2 stabilization coefficients. This metric leverages a kernel to extract the structural information of images, rather than focusing on single-pixel errors. This leads to a more robust metric compared to other alternatives like MSE and PSNR.

Denoising The first set of experiments focuses on the image-denoising task (as shown in Figure 2). Examining the results presented in Table 1, NAFNet and Restormer achieve the highest performance on all the tested datasets. Nevertheless, ViTCA obtains comparable results. As expected, latent models, namely LatentViTCA and LatentNAFCA, exhibit a decrease in performance due to the downsizing of the input. Upon close examination, we notice a decrease of $\approx 5\%$ in denoising efficiency for LatentViTCA, compared to ViTCA, and a slightly larger decrease of $\approx 7\%$ for LatentNAFCA. We emphasize that a task utility downgrade is expected for latent architectures, as they are designed to trade restoration performance for computational and memory efficiency. In the Efficiency Results Section we comment on this tradeoff by analyzing the resources required by all models.

Deblurring Results on the deblurring task (as shown in Figure 3), follow a similar trend to denoising, as shown in Table 2. In particular, NAFNet and Restormer still achieve the best results. ViTCA follows as the third best alternative, although the performance gap is higher than the denoising task. Compared to ViTCA, latent models exhibit a greater decrease in performance of $\approx 13\%$ for LatentViTCA and

Table 3: Training memory requirements tests. Results are reported once the memory has stabilized after an initial warm-up phase, measured in GB. If the configuration requires more than the available 80 GB of VRAM, the measurement is not available and is indicated by "–". The best results are highlighted in bold, while the second best are underlined.

Image sizes	32×32			64×64			128×128		
Batch sizes	8	16	32	8	16	32	8	16	32
NAFNet	4.1	4.1	4.2	4.2	8.2	8.8	8.7	19.6	35.0
Restormer	4.4	4.4	8.5	8.8	16.8	32.9	32.8	65.6	–
ViTCA	8.7	16.9	33.8	33.8	65.6	–	–	–	–
LatentViTCA	0.5	1.1	1.2	2.2	4.4	4.7	8.7	16.9	17.4
LatentNAFCA	**0.5**	**0.5**	**1.1**	**1.1**	**2.1**	**4.1**	**4.1**	**8.2**	**16.4**

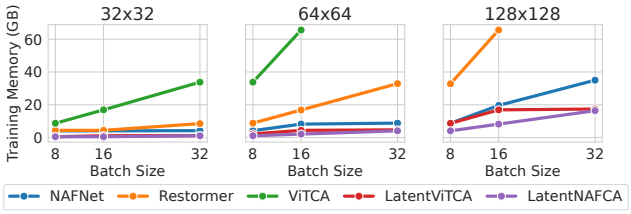

Figure 4: Training memory trend across configurations.

$\approx 15\%$ for LatentNAFCA. This gap is less prominent in real-world datasets – GoPro and RealBlur – which are influenced by less critical corruptions than synthetic datasets.

Efficiency Results

This section focuses on the efficiency analysis of the proposed solution, LNCA, compared with the existing IR architectures. We measure and compare four metrics: training memory, inference memory, training latency, and inference latency. These metrics summarize the resource usage and computational requirements of a generic deep learning architecture. The objective of LNCA is to reduce these metrics with respect to other NCA-based models. In this context, our primary comparison is with ViTCA, which belongs to the NCA family. However, we will also extend our tests to NAFNet and Restormer to ensure a more comprehensive understanding of the results.

To enable testing on a wider range of input sizes, the following experiments were performed on a cloud service using an Nvidia A100 GPU with 80GB of VRAM, which has been the main limiting factor for NCA architectures.

Training memory requirement This metric (Table 3 and Figure 4) is calculated as the highest VRAM requirement for each model at a given input resolution, typically reached after an initial warm-up period and then held constant throughout training. In the case of latent models, we consider the maximum between the AE and NCA training phases.

Starting from the lowest image resolution tested, namely $32 \times 32 \times 3$ with a batch size of 8, we notice a reduction

Table 4: Inference memory requirements tests. Results are reported once the memory has stabilized after an initial warm-up phase, measured in GB. The best results are highlighted in bold, while the second best are underlined.

Image sizes	32×32			64×64			128×128		
Batch sizes	8	16	32	8	16	32	8	16	32
NAFNet	1.1	1.1	1.1	1.1	1.1	1.6	1.6	7.3	6.3
Restormer	1.3	1.3	1.3	1.6	2.1	2.1	2.1	4.1	8.3
ViTCA	1.5	2.6	5.1	5.1	10.3	12.4	12.4	17.5	36.1
LatentViTCA	**0.2**	**0.2**	0.4	0.4	0.8	1.6	1.6	4.6	5.1
LatentNAFCA	0.3	0.3	**0.3**	**0.3**	**0.3**	**0.3**	**0.3**	**0.5**	**0.5**

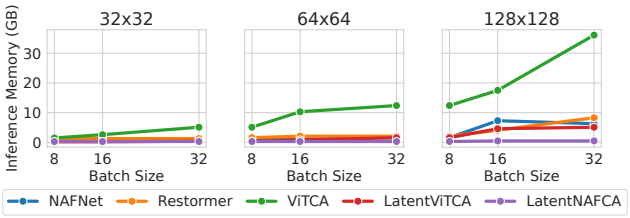

Figure 5: Inference memory trend across configurations.

of $\approx 94\%$ for both our solutions compared to ViTCA. This reduction remains nearly constant across all input configurations. Furthermore, considering the minimum memory requirement of ViTCA, i.e., 8.7GB, we achieve a 16x improvement in the training input size by using its latent counterpart. Interestingly, the latent models are also up to $\approx 88\%$ more efficient than the classical solutions, which should have a natural advantage due to their non-recurrent structure. Overall, we can see that our solutions (and NAFNet) cope well with the increase in batch size, keeping the requirements almost stable, while the other models turn out to be less scalable.

Inference memory requirement This metric (Table 4 and Figure 5) is computed as the highest VRAM requirement for each model during the inference of 10 batches, across various input sizes. As expected, the inference requirements are significantly lower than the training ones, due to the absence of backpropagation. Nevertheless, the results are aligned with those concerning training memory, with latent models leading in almost every configuration. At the smallest input tested ($8 \times 32 \times 32 \times 3$), we observe an $\approx 87\%$ and $\approx 80\%$ memory reduction from our solutions compared to ViTCA. As the input increases, ViTCA requirements tend to diverge, while the other models show a gentler curve, with LatentNAFCA being exceptionally efficient.

Training latency In this experiment, we adopt a dummy dataset of 1000 noisy samples (80% training, 20% validation) and perform a 10-epoch training. We compute the average training latency discarding the values of the initial warm-up epoch. For latent models, we sum the AE and NCA training times to get a fair estimate of total training time.

Table 5: Training latency tests performed on a dummy dataset of 1000 samples (80 − 20 split). The results are the average of 10 training epochs (after the initial warm-up), measured in seconds. If the configuration requires more than the available 80 GB of VRAM, the measurement is not available and is indicated by "–". The best results are highlighted in bold, while the second best are underlined.

Image sizes	32×32			64×64			128×128		
Batch sizes	8	16	32	8	16	32	8	16	32
NAFNet	25.1	13.3	7.9	22.8	13.0	9.0	<u>28.8</u>	<u>19.0</u>	<u>15.7</u>
Restormer	<u>21.7</u>	<u>11.9</u>	7.9	25.2	18.9	16.8	59.9	57.1	–
ViTCA	21.7	18.5	17.5	69.9	61.6	–	–	–	–
LatentViTCA	**14.9**	**8.9**	**4.9**	**16.6**	<u>11.8</u>	<u>8.4</u>	30.0	21.8	16.4
LatentNAFCA	24.5	14.5	<u>7.2</u>	<u>17.1</u>	**11.4**	**7.4**	**21.2**	**14.8**	**10.2**

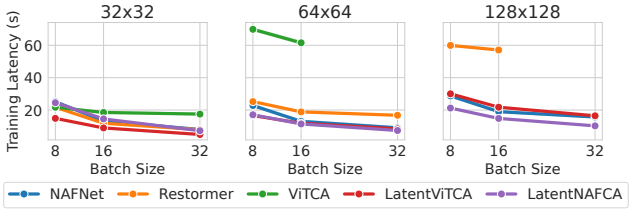

Figure 6: Training latency trend across configurations.

As expected, with a fixed-size dataset, increasing the batch size leads to an improvement in the overall training time for all models (Table 5 and Figure 6). At the lowest resolution ($32 \times 32 \times 3$), the difference between ViTCA and the latent models is negligible, although the decreasing slope is less pronounced in the former. Increasing the resolution, we notice the degradation in performance of ViTCA and, subsequently, of Restormer, while the remaining models keep stable timings. In our view, the point at which performance begins to decline is closely tied to the memory usage of the models. Specifically, processing latency deteriorates as the model starts to saturate the GPU VRAM. Overall, latent models achieve up to $\approx 80\%$ reduction in training latency compared to ViTCA, making them comparable to the NAFNet.

Inference latency Regarding the inference latency, tests are performed using a single batch as input and averaging over 10 iterations. Similarly to the training latency, we discard the first processing of each model as a warm-up.

We expect our solutions and ViTCA to be significantly slower than classical models due to the NCA computation. In fact, we fix these models to perform $t = 64$ NCA processing steps instead of the single pass used by the non-NCA models. Looking at the results (Table 6 and Figure 7), we can see that our solutions and NAFNet maintain almost stable performance across all resolutions, while the other models diverge once a memory breakpoint is reached. As previously mentioned, we believe this trend is directly related to VRAM usage. At the highest resolution ($128 \times 128 \times 3$), where

Table 6: Inference latency tests performed on a single batch. The results are the average of 10 inference steps (discarding the initial warm-up), measured in seconds. The best results are highlighted in bold, while the second best are underlined.

Image sizes	32×32			64×64			128×128		
Batch sizes	8	16	32	8	16	32	8	16	32
NAFNet	**0.25**	**0.24**	**0.25**	**0.24**	**0.24**	**0.24**	**0.24**	**0.25**	**0.25**
Restormer	<u>0.56</u>	<u>0.53</u>	<u>0.54</u>	<u>0.54</u>	<u>0.55</u>	<u>0.58</u>	<u>0.58</u>	0.68	0.93
ViTCA	1.47	1.42	1.45	1.43	1.46	2.00	2.01	2.96	5.17
LatentViTCA	1.45	1.45	1.46	1.43	1.43	1.41	1.43	1.44	1.42
LatentNAFCA	0.59	0.60	0.60	0.59	0.60	0.59	0.59	<u>0.60</u>	<u>0.59</u>

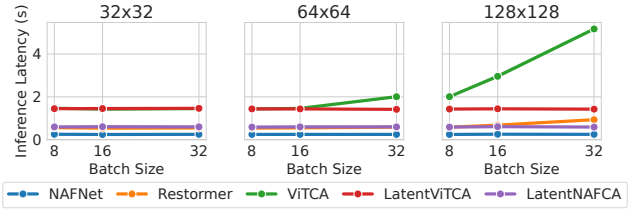

Figure 7: Inference latency trend across configurations.

memory requirements are demanding, our latent models are respectively up to $\approx 72\%$ and $\approx 89\%$ faster than ViTCA, with LatentNAFCA also $\approx 37\%$ faster than Restormer.

Conclusions

NCAs are fascinating models capable of learning complex system behavior from simple local interactions. Yet, their practical application has been severely limited by their computational constraints. This work focused on mitigating this critical challenge, providing an NCA model, called LNCA, with low memory requirements and latency bottlenecks. LNCA is a novel deep learning-based architecture that moves the computationally demanding NCA computation from the input space to a lower-dimensional manifold. This manifold is tailored by an autoencoder to extract the essential information for the NCA to solve the task. We validated LNCA in the field of image restoration, specifically addressing denoising and deblurring tasks. The experiments showcased an effective reduction in the computational requirements of NCA. With respect to the SotA, LNCA achieved up to a $16\times$ improvement in the maximum input size, and up to a $\approx 94\%$ and $\approx 80\%$ reductions in memory requirements and processing latency, respectively, with the drawback of an overall degradation in restoration performance of $\approx 10\%$. In conclusion, LNCA successfully addressed some of the inherent limitations of NCA models, providing a flexible and practical solution for the integration of cellular automata into deep learning architectures.

References

Anaya, J. and Barbu, A. (2018). RENOIR – a dataset for real low-light image noise reduction. *Journal of Visual Communication*

and Image Representation, 51:144–154.

Banham, M. and Katsaggelos, A. (1997). Digital image restoration. *IEEE Signal Processing Magazine*, 14(2):24–41.

Chen, C., Chen, Q., Xu, J., and Koltun, V. (2018). Learning to see in the dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, L., Chu, X., Zhang, X., and Sun, J. (2022). Simple baselines for image restoration. In *European Conference on Computer Vision*, pages 17–33. Springer.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 933–941. JMLR.org.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. `http://www.deeplearningbook.org`.

Hendrycks, D. and Gimpel, K. (2023). Gaussian error linear units (gelus).

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Loshchilov, I. and Hutter, F. (2017). Sgdr: Stochastic gradient descent with warm restarts. In *ICLR (Poster)*. OpenReview.net.

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). Growing neural cellular automata. *Distill*. https://distill.pub/2020/growing-ca.

Nah, S., Kim, T. H., and Lee, K. M. (2017). Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*.

Park, T., Zhu, J.-Y., Wang, O., Lu, J., Shechtman, E., Efros, A., and Zhang, R. (2020). Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33:7198–7211.

Rendell, P. (2002). *Turing Universality of the Game of Life*, pages 513–539. Springer London, London.

Rim, J., Lee, H., Won, J., and Cho, S. (2020). Real-world blur dataset for learning and benchmarking deblurring algorithms. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 184–201, Cham. Springer International Publishing.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Sudhakaran, S., Grbic, D., Li, S., Katona, A., Najarro, E., Glanois, C., and Risi, S. (2021). Growing 3d artefacts and functional machines with neural cellular automata.

Tesfaldet, M., Nowrouzezahrai, D., and Pal, C. (2022). Attention-based neural cellular automata. *Advances in Neural Information Processing Systems*, 35:8174–8186.

Variengien, A., Nichele, S., Glover, T., and Pontes-Filho, S. (2021). Towards self-organized control: Using neural cellular automata to robustly control a cart-pole agent.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

von Neumann, J. (1951). The general and logical theory of automata. In Jeffress, L. A., editor, *Cerebral Mechanisms in Behaviour*. Wiley.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Zamir, S. W., Arora, A., Khan, S., Hayat, M., Khan, F. S., and Yang, M.-H. (2022). Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739.