Self-Organizing Maps for the Reconstruction of Images in Pixel Permuted Image Stacks

Connor Price, David Kott, Chris Peterson, Michael Kirby
Department of Mathematics, Colorado State University, Fort Collins, CO 80523
Email: {Connor.Price, David.Kott, Christopher2.Peterson, Michael.Kirby}@Colostate.edu

Abstract

A color digital photograph, of resolution $a \times b$, is typically stored as an $a \times b \times 3$ array. The vector of length 3, sitting at a pixel, encodes its color in terms of intensity measurements of the red, green, and blue color bands. We call this length 3 vector a "pixel pole". Suppose we are given the ab individual pixel poles as a jumbled collection of length 3 vectors and we wish to reconstruct the image, with no advanced knowledge concerning its content. In other words, we are given ab color squares and we wish to "solve the pixel puzzle" meaning we want to reconstruct the original unknown image. As one can imagine, this is a difficult problem. In this paper, we show how to rebuild the images in a stack of N distinct $a \times b$ color digital images from the ab stacked pixel poles. More precisely, this paper shows how to use "Self Organizing Maps" (SOMs) to algorithmically reconstruct, unsupervised, a stack of distinct $a \times b$ color images from the collection of $1 \times 1 \times 3N$ stacked pixel poles using no apriori information about the original images. We evaluate the accuracy of the reconstructions as a function of N, we determine the effectiveness of the algorithm when the individual images are corrupted by noise, and we assess the model's performance when pure noise images are included in the stack.

1 Introduction

In this study, we consider the problem of reconstructing the images in a stack of digital images from the pixel data in the stack. We start with a collection of N distinct color photographs, all with the same resolution. Considering color channels, each photograph is represented by an $a \times b \times 3$ array. If we were to stack the digital photographs, they would combine to make an array, A, of size $a \times b \times 3N$. We can think of A as consisting of an arrangement of ab vectors of length $3N$. Our problem is to reconstruct A from the collection of ab unlabeled length $3N$ vectors with no apriori knowledge about the content of the stack. To tackle this challenge, we employ self-organizing maps (SOMs), an unsupervised learning algorithm well-suited for uncovering geometric structure within a data set.

SOMs have been used in many contexts for data visualization and clustering [5]; see also [10] for a more recent review. In applications, SOMs have proven useful in a wide array of examples including the modeling of water resources [4], pattern recognition in satellite imagery [12], and finance [2]. Further, SOMs have been proposed for a variety of biological applications including interpreting gene expression data [13] and applying it to the molecular classification of cancer [1]. More recently, SOMs have been applied in broader geometric settings, including the Grassmannian [8] and the flag manifold [7]; see [9] for additional details on the application of the flag manifold for the analysis of large data sets. The geometry that the SOM uncovers is a reflection of the overall correlation expected between values in neighboring pixels. Thus, we incorporate a sense of local smoothness in our reconstruction process with the aim of minimizing abrupt transitions and discontinuities between adjacent pixels.

Proceedings of the 1st Conference on Topology, Algebra, and Geometry in Data Science(TAG-DS 2025).

Recent advancements in image reconstruction have largely focused on 'visual jigsaw puzzles,' where the atomic units are image patches or tiles rather than individual pixels. For instance, Paumard et al. [11] introduced 'Deepzzle,' a method utilizing deep graph learning to reassemble square fragments, relying on learned visual features within each 96×96 tile to bridge eroded gaps. Similarly, Liu et al. [6] proposed 'Jigsaw Puzzles with Diffusion Vision Transformers' (JPDVT), which employs conditional diffusion models to reassemble and inpaint missing patches in both spatial and temporal contexts.

A fundamental distinction, however, lies in the granularity of the problem. Both Deepzzle and JPDVT operate on patches containing inherent spatial structure (edges, textures) that discriminative or generative models can learn. In contrast, our SOM-based approach addresses the challenge of pixel-level permutation. At this resolution, individual data points lack spatial context or texture. Consequently, our method cannot rely on intra-patch features but instead leverages the statistical coherence across a stack of images to recover the topological structure—a problem where traditional feature-based puzzle solvers would struggle due to the lack of spatial information per element.

The primary objectives of this project are as follows:

1. Evaluate Unshuffling Accuracy: We assess the effectiveness of SOMs in accurately reconstructing the shuffled images.
2. Noise Resilience: We investigate the model's capability to reconstruct the stack of images when a percentage of the images are corrupted by noise.
3. Noise Identification: We examine the model's ability to identify and isolate the noisy images within the stack.
4. Lattice Flexibility: We assess the adaptability of the model's lattice structure in accommodating various distinct configurations.

The positive results found in pursuing these objectives demonstrate the robustness and applicability of SOMs in handling real-world image reconstruction tasks.

2 Methods

We utilized the ImageNet dataset [3], which contains over 1.2 million images categorized into 1,000 classes. This diverse dataset ensured a variety of images without overlapping borders, facilitating a robust evaluation of the model's performance. To build our stacks, we selected images from the dataset with a resolution greater than 64x64 and then applied random cropping. Pixel colors are encoded in a vector of three numbers, with each entry an integer between 0 and 255.

2.1 Flattening and Shuffling

The pixels in the stack were shuffled in a way that each image in the stack had the same pixel permutation applied. This step simulated the challenge of reconstructing the original images from a shuffled stack. The bottom row of Figure 1 displays the challenge of turning what appears to be flavored TV static back into the original image (the top row).

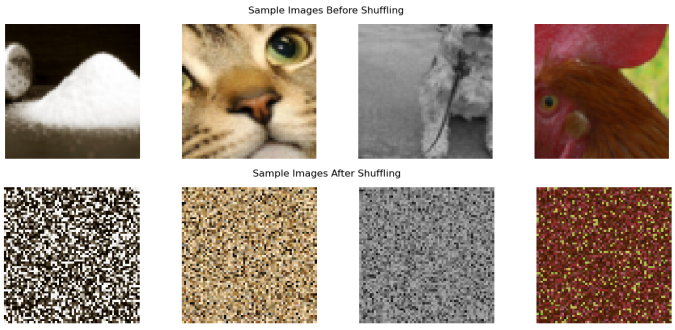


Figure 1: A sample stack of four images in the top row and an example of that stack permuted. Our goal is to take the bottom row as input and return the top row of images.

Before applying the SOM algorithm to the pixel-permuted stack of images, we flatten the length and width dimensions of the stack. For images with a resolution of 64 by 64, this process results in 4,096 vectors. The SOM algorithm aims to map neurons to this set of vectors. A goal of the SOM algorithm is to map data from the input space to a space with a predetermined topology. This topology can be something as simple as a lattice or something more complex, such as a Klein bottle. The topology is implemented through the distance matrix used for weighted updates in line 10 of our adapation of the SOM algorithm [5] (see Algorithm 1). Through testing, we show that the neurons don't need to be fixed to match the number of input samples and can actually be significantly more or less while still maintaining their ability to restore the original image.

For our use case, we utilize a two-dimensional lattice as the target topology, which naturally agrees with the topological layout of the pixels in an image. This choice allows us to treat each pixel position as a point, and the lattice represents the sought-after, organized reconstruction of these points (see Algorithm 1). By treating all pixel positions as an orderless list of vectors, we can process all permutations of the shuffled pixels as identical inputs. This approach allows us to generalize a successful restoration of an image stack to all possible permutations of shuffling.

2.2 Algorithms for Self-Organizing Maps (SOMs)

Algorithm 1 SOM Algorithm for Image Unshuffling

1: **Input:** Learning rate η, grid size R, number of epochs γ, shuffled images X, width parameter μ
2: **Initialize:** $C =$ lattice of size $R \times R$
3: $\alpha = e^{\frac{1}{\gamma}\mu}$
4: $D_a^b =$ distance between C^a and C^b on the lattice coordinates
5: width = μ
6: **for** each iteration $t = 1, 2, \ldots, \gamma$ **do**
7: width = width / α
8: **for** each pixel position $l = 1, 2, \ldots, R^2$ **do**
9: $m = \arg\min_i \{||X_l - C_i||_2\}$
10: $\beta_j = e^{-\frac{D_j^{m\,2}}{\text{width}^2}}$
11: $C_{k+1}^j = C_k^j + \eta\beta_j(X_l - C_k^j)$
12: **end for**
13: **end for**
14: **Output:** Optimized neurons C

Algorithm 2 Image Processing with Self-Organizing Map (SOM)

1: **Input:**Number of images in the stack N, resolution of each image R
2: x = N images, $x \in \mathbb{R}^{R\times R\times 3\times N}$
3: x = flatten(x, first dimension), $x \in \mathbb{R}^{R^2\times 3\times N}$
4: x = flatten(x, last dimension), $x \in \mathbb{R}^{R^2\times 3N}$
5: x = shuffle(x, first dimension)
6: y = SOM(X = x, η = .2, R = R, γ = 400), $y \in \mathbb{R}^{R^2\times 3N}$
7: y = split(y, first dimension), $y \in \mathbb{R}^{R\times R\times 3N}$
8: y = split(y, last dimension), $y \in \mathbb{R}^{R\times R\times 3\times N}$
9: **Output:** Reconstructed stack of images y

Starting off, we align the resolution of our lattice with the resolution of the reconstructed images. Here, we treat each pixel position as a neuron within the SOM algorithm, thereby ensuring that each neuron is uniquely mapped to an input vector.

The SOM training process involves several key steps. We begin by initializing a lattice with a resolution of 64 by 64 neurons, where each neuron is a vector corresponding to a pixel's position and the color values of all images within the stack at that pixel location. This initialization sets up the grid that will be used for mapping and reconstructing the images.

Image sub-sampling and pre-processing, before SOM is performed, takes place in Algorithm 2, lines 2 through 5. During a training iteration, a pixel position is chosen (only once per epoch) and the algorithm identifies the closest neuron. The winning neuron and its neighboring neurons then update their weights to better match the input vector proportional to the distance described in the target topology. For a "winner" neuron to be identified, we have used the Euclidean metric. Thus, the winning neuron is identified via:

$$\arg\min_i \{||X_l - C_i||_2\} \tag{1}$$

This weight adjustment process is crucial as it ensures that the SOM neurons gradually learn to represent the spatial relationships between pixel values, thereby projecting the inputs onto the desired topology. The width variable in Algorithm 1 dampens the amount a neuron can update as epochs increase. We make our neuron weight updates with the following formalization:

$$C^j_{k+1} = C^j_k + \eta\beta_j(X_l - C^j_k) \tag{2}$$

Instead of allowing a single input to update one or all neurons, a vector β is used to update neurons in proportion to their distance from the winner neuron on the target topology. Over time, the weights are adjusted through repeated training epochs, effectively learning the underlying structure of the data. Once training is complete, the trained SOM weights are mapped to their original position on the lattice (variable C on line 2 of Algorithm 1), resulting in the reconstruction of the images.

We observe that the SOM algorithm attempts to maximize the "smoothness" of each pixel relative to its neighbors. For example, given a picture of a boat on the ocean, the SOM will try to keep dark blue values (ocean) and bright blue values (sky) grouped together. The algorithm struggles with borders, such as the outline of a boat, which is why we use a stack of images. We hypothesize that by combining the net effect of multiple images, a sharp border between objects in a single photo will be counterbalanced by the smooth patches of most of the other photos at the same pixel positions.

2.3 Performance Analysis

Four primary tasks were used to assess the versatility of the SOM algorithm:

1. **Image Reconstruction**: The effectiveness of SOMs to accurately reconstruct the shuffled images was measured using the absolute mean difference in value between each pixel position.
2. **Noise Resilience**: The model's capability to reconstruct the stack of images when a percentage of the images were corrupted by noise was evaluated by adding varying levels of uniform noise to the images and assessing the reconstruction quality.
3. **Noise Identification**: The model's ability to identify and isolate noisy images within the stack was assessed using a noise detection algorithm that flagged images based on a smoothness metric.
4. **Lattice Flexibility**: This evaluation helps determine the robustness of the SOM algorithm when we vary the underlying lattice, especially in scenarios where image dimensions and layouts vary. We will explore different lattice configurations, such as rectangular and custom grids, to identify the optimal structure for our image reconstruction tasks.

Image Reconstruction: To evaluate unshuffling accuracy, we computed the mean absolute error between the true and converged pixel values. This metric measures the displacement of pixels from their original locations, quantifying the model's precision in reassembling the image stack.

Noise Resilience: To examine the robustness of smoothness as a means of restoration, we test the SOM algorithm on a stack of images with varying amounts of noise added. We select n images from our stack of N and denote them as $I^1, I^2, \ldots, I^n$. For each image I^k ($k = 1, 2, \ldots, n$), we alter the RGB values at pixel position (x, y) with a randomly generated 3D vector $\mathbf{v}^k_{x,y}$, where $\mathbf{v}^k_{x,y}$ is drawn from a uniform distribution between 0 and 255 for each of the three color channels (R, G, B). Through this experiment, we aim to demonstrate the robustness and applicability of SOMs in handling image reconstruction tasks in the presence of noise.

The success of converged reconstruction was evaluated to determine the extent to which the image stack can be corrupted while still allowing for accurate reconstruction. Destroying only a few images is expected to have a minimal impact on the performance of the SOM. With a sufficient number of untouched images, it should be possible to reconstruct the non-noisy images despite the presence of the noisy images.

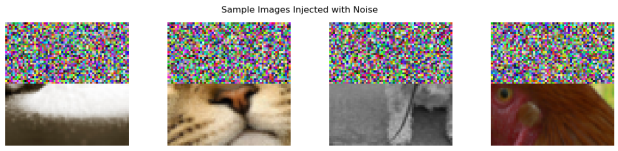


Figure 2: A sample stack of four images in the top row injected 50% with noise prior to shuffling

Noise Identification: By employing a metric for smoothness, we can assign a smoothness score to each image and identify which images are likely corrupted. Our metric for measuring the smoothness of an image I^k, denoted as S^k, is defined by calculating the average absolute difference between each pixel and its immediate neighbors. We extend this definition to consider the three color channels (R, G, B) and handle corner cases.

$$S^k_{(x,y)} = \frac{1}{\mathrm{T}} \sum_{f=1}^{3} \left(\sum_{i,j \in \{-1,0,1\}} \frac{\left|\left| I^k_{(x,y)} - I^k_{(x+i,y+j)} \right|\right|_1}{3} \right) \tag{3}$$

In the formula above, $I^k_{(x+i,y+j)}$ is the intensity of the neighboring pixel, and T is the number of valid neighbors for a pixel position. Thus, T equals 8 for most positions but 3 for corners and 5 for edges. Summing over the three color channels guarantees that the total smoothness metric S_k accounts for the entire image, effectively ignoring out-of-bound areas in the calculation.

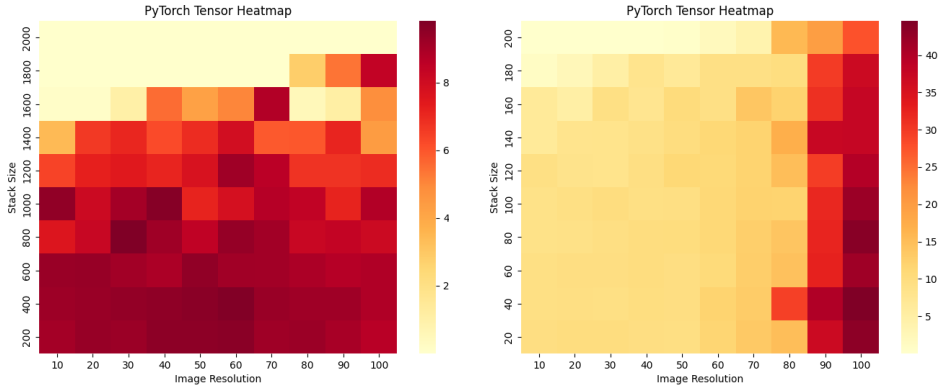
Lattice Flexibility: Our lattice flexibility evaluation was designed to assess the impact of different neuron configurations on the overall reconstruction and accuracy of the SOM. To evaluate neuron flexibility, we experimented with different lattice configurations, including standard square lattices, expanded lattices, and unusually shaped regions. Each configuration was tested on the same stack of images to ensure consistency in the evaluation process. The following configurations were explored:

1. **Standard Square Lattice**: A standard square lattice configuration (e.g., 64x64) was used as a baseline to assess the model's reconstruction accuracy under default settings.
2. **Expanded Lattice**: The lattice was expanded to a larger size (e.g., 128x128) to evaluate performance when more neurons exist than pixel positions.
3. **Shrunk Lattice**: The lattice was shrunk to a smaller size (e.g., 32x32) to evaluate performance when fewer neurons exist than pixel positions.
4. **Non-Standard Shape**: A non-rectangular configuration was tested to determine the effectiveness of non-square lattices in handling images with varying dimensions and layouts.

3 Results and Discussion

A key consideration in our experimental design was determining the minimum number of images necessary for a successful reconstruction. With the image resolution fixed at 64x64, we assumed the primary constraint would be the stack size, as limited by GPU memory. However, a stack of approximately 100 images yielded reconstruction results comparable to those achieved with much larger stacks, even though our GPUs could accommodate up to $\sim$650,000 images. This reveals that the primary constraint is not computational resources but the intrinsic information needed by the algorithm. Beyond a certain threshold, adding more images does not provide additional structural information to aid the unshuffling process.

All experiments were conducted on an NVIDIA DGX-2 system using a single GPU. The training process for a standard configuration, consisting of a 100-image stack at 64x64 resolution over 400



(a) (b)

Figure 3: Image Resolution vs Stack Size

epochs, was consistently completed in approximately 4.8 minutes (289 seconds). This execution time remained stable across various experimental conditions.

3.1 Evaluate Unshuffling Accuracy

One question we explored was the effect of the images' resolution, in the stack of images, on the reconstruction of an image. To display our results, we created heatmaps with image resolution on the X-axis and image count in stacks on the Y-axis. Each pixel value represents the median score of 16 runs using our metric. Low values indicate that the converged image only slightly differed from the original, whereas large values signify a significant difference and indicate a failed reconstruction.

We generated two heatmaps to display a wider range of stack sizes. Figure 3a (a) covers stack sizes from 200 to 2,000 and has a color bar ranging from 0 to 10. The more granular heatmap, Figure 3b (b), uses stack sizes ranging from 10 to 200 and a color scale from 0 to 45. Different color scales were used for the two figures to ensure clarity; a single scale from 0 to 45 would render Figure 3a almost entirely one color.

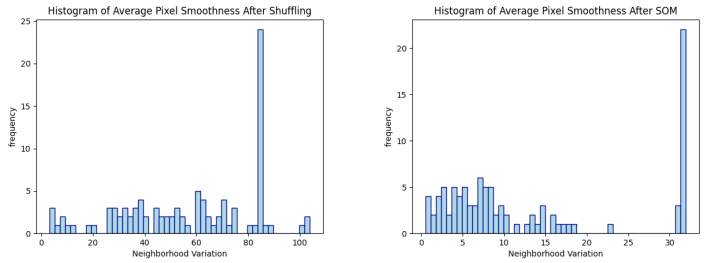
In our larger stack of images, all models converged to a point of clear recognition. Notably, at the top of Figure 3a, an almost white strip indicates near-perfect reconstruction.

For the smaller stack size, we observed a clear increase in difficulty for images with a resolution of 90 by 90 or greater. This test also revealed instances of failed reconstructions, as illustrated in Figure 4.

Figure 4: Examples of Failed Reconstructions

3.2 Noise Resilience

Our method of utilizing SOMs for reconstruction relies on the smoothness of an area throughout many images. To test robustness, we removed some smoothness. To achieve this, we started with a stack of 100 images, each with a resolution of 64 by 64 pixels, and transformed a portion into complete noise. This noise was generated using a random uniform distribution. By varying the amount of noisy images in increments of 10, we observed the relationship between the number of corrupted images and performance. The effect of noise appears to be linearly proportional to its increase in error over the range considered (Figure 5).



(a) Distribution of average pixel value differences after shuffling.

(b) Distribution of average pixel value difference after SOM reconstruction.

Figure 7: Distribution of average pixel value differences

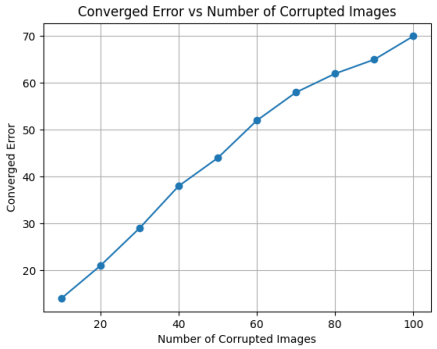


Figure 5

When comparing the converged images between an untouched stack and a stack with 25 noisy images, it is apparent from Figure 6 that the increase in error is reflected as a blurring of the original image.

Figure 6: Despite our original stack of images (left) having 25% noise, the SOM was able to converge (right) to the original stack with only slight blurring.

3.3 Noise Identification

One area of interest was using SOMs as a method for noise identification. We observed that converged images have smooth color patches, whereas noise does not. Our test was conducted with a stack of 100 images, each with a resolution of 64 by 64, where 25 of these images were turned into complete noise. After shuffling, we noticed some consistent coloring, as shown in the bottom row of Figure 1. Figure 7a demonstrates that, despite some pixel color consistency in un-noised shuffled images, separating the 25 noised images using a smoothness threshold would not perfectly distinguish real from noise images within the shuffled stack.

When applying SOM to the partially corrupted stack, we observed a more apparent separation between noise and real images. Any image with a neighborhood variation greater than 25 was identified as a corrupted image in the stack (Figure 7b).

3.4 Lattice Flexibility

We used simple scaling as our starting point to examine how increasing or decreasing the neuron grid size affects the converged resolution. A stack of 100 images, each with a resolution of 64 by 64 pixels, was used as input for every experiment.

Lowering image resolution can be achieved by using fewer than 64 neurons for the lattice creation on line 2 of Algorithm 1. This initialization creates a grid coarser than the input stack, forcing a change in the behavior of the minimization step in Line 9. Because there are more input vectors in X than available neurons in C, the search for a winning neuron results in a many-to-one mapping. Consequently, single neurons effectively become the centroid for clusters of multiple input pixels, resulting in a downsampled approximation of the original. A neuron resolution of 32 by 32 was used, and the resulting converged image can be seen in Figure 8 part (a). The model can still converge to a recognizable image.

Another test involved setting the neuron lattice to 128 by 128 to evaluate its ability to upscale image resolution. The aim was to determine whether higher resolution was achieved by simply converging multiple neurons to a single pixel position or if some form of object boundary smoothing was achieved. Comparing the 32 by 32 stack of images, Figure 8 part (a), with Figure 8 part (b), the high-resolution reconstructed sample, shows that the model blurs the boundaries of objects to create what appears to be a sharper image in comparison.

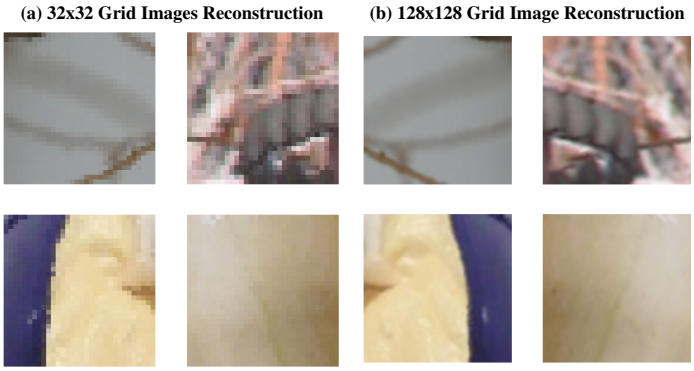


Figure 8

The most straightforward topology for SOMs to recover is a grid, the same topology as the input. However, SOMs are flexible to a topology that differs from the input. As an illustrative example, we applied the SOM algorithm to a topology representing a more complex space. We used face images to test different topologies, each forming one of the letters in "PAL". The configuration of neurons to form each desired letter was executed using the same stack as in previous experiments, with the exception of replacing the top pictures in the stack with images of the two faces. Despite the significant difference from the input topology, each of the runs successfully reproduced the face images to a recognizable degree, as shown in Figure 9.

Figure 9: Demonstration of lattice flexibility. The SOM algorithm successfully reconstructs face images from the stack onto topologies shaped like the letters 'P', 'A', and 'L'.

4 Conclusion

In this study, we demonstrated the effectiveness of SOMs in reconstructing pixel-shuffled image stacks, even in the presence of noise. Our results indicate that SOMs can accurately unshuffle images, identify noisy images, and maintain robustness across various lattice configurations. The ability of SOMs to isolate and identify noise within image stacks is noteworthy. This capability suggests that SOMs can be leveraged for feature selection in broader applications. By identifying and isolating noisy or irrelevant features, SOMs can enhance the performance of machine learning models, leading to more accurate and reliable outcomes.

Our experiments with different lattice configurations highlight the flexibility of SOMs in adapting to various structures. This adaptability makes SOMs a tool for various image processing and data visualization tasks. Overall, our findings underscore the potential of SOMs in handling complex image reconstruction challenges and their applicability in feature selection and noise resilience. Future work could explore the integration of SOMs with other machine learning techniques to enhance their capabilities further and extend their use to different domains.

Acknowledgments: This material is based upon work supported by the National Science Foundation under Cooperative Agreement No. 2428052. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] David G Covell, Anders Wallqvist, Alfred A Rabow, and Narmada Thanki. Molecular classification of cancer: unsupervised self-organizing map analysis of gene expression microarray data. *Molecular cancer therapeutics*, 2(3):317–332, 2003.

[2] Guido Deboeck and Teuvo Kohonen. *Visual explorations in finance: with self-organizing maps*. Springer Science & Business Media, 2013.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[4] Aman Mohammad Kalteh, Peder Hjorth, and Ronny Berndtsson. Review of the self-organizing map (som) approach in water resources: Analysis, modelling and application. *Environmental Modelling & Software*, 23(7):835–845, 2008.

[5] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[6] Jinyang Liu, Wondmgezahu Teshome, Sandesh Ghimire, Mario Sznaier, and Octavia Camps. Solving masked jigsaw puzzles with diffusion vision transformers. In *CVPR*, pages 23009–23018, 2024.

[7] Xiaofeng Ma, Michael Kirby, and Chris Peterson. Self-organizing mappings on the flag manifold. In *International Workshop on Self-Organizing Maps*, pages 13–22. Springer, 2019.

[8] Xiaofeng Ma, Michael Kirby, Chris Peterson, and Louis Scharf. Self-organizing mappings on the grassmannian with applications to data analysis in high dimensions. *Neural Computing and Applications*, 2019.

[9] Xiaofeng Ma, Michael Kirby, and Chris Peterson. The flag manifold as a tool for analyzing and comparing sets of data sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 4185–4194, 2021.

[10] Dubravko Miljković. Brief review of self-organizing maps. In *2017 40th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1061–1066. IEEE, 2017.

[11] Marie-Morgane Paumard, David Picard, and Hedi Tabia. Deepzzle: Solving visual jigsaw puzzles with deep learning and shortest path optimization. *IEEE TIP*, 29:3569–3581, 2020.

[12] Anthony J Richardson, C Risien, and Frank Alan Shillington. Using self-organizing maps to identify patterns in satellite imagery. *Progress in Oceanography*, 59(2-3):223–239, 2003.

[13] Pablo Tamayo, Donna Slonim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S Lander, and Todd R Golub. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Sciences*, 96(6): 2907–2912, 1999.

TAG-DS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: The abstract and introduction claim to evaluate the accuracy of using Self-Organizing Maps (SOMs) for image reconstruction, assess their resilience to noise, their ability to identify noise, and their flexibility with different lattice structures. The rest of the paper provides experiments and results that directly support each of these claims.

2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: The paper clarifies that the algorithm's performance is not constrained by computational resources. The number of images required to achieve acceptable accuracy is significantly smaller than the memory capacity of standard GPUs. The primary resource requirement is a system capable of handling a moderately sized image stack (for a 64x64 resolution), which is well within the capacity of most modern GPUs.

3. **Theory assumptions and proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [NA]
   Justification: This paper is empirical in nature. It proposes and experimentally validates an algorithmic approach for image reconstruction. All theory is already established and cited.

4. **Experimental result reproducibility**
   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?
   Answer: [Yes]
   Justification: The paper describes the dataset used (ImageNet), preprocessing steps (random cropping, flattening, shuffling), the core SOM algorithm with key hyperparameters (learning rate $\eta = 0.2$, epochs $\gamma = 400$), and the specific metrics for evaluation (mean absolute error, smoothness score). This level of detail is sufficient for another researcher to replicate the experiments.

5. **Open access to data and code**
   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in the supplemental material?
   Answer: [Yes]
   Justification: The code can be found on the following GitHub:
   https://github.com/cprice44/som-image-unshuffling/tree/main

6. **Experimental setting/details**
   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?
   Answer: [Yes]
   Justification: The paper specifies the dataset, image resolutions, stack sizes, learning rate, and number of epochs. The core of the method is the SOM algorithm itself, which is detailed in Algorithm 1, making the experimental setup clear.

7. **Experiment statistical significance**
   Question: Does the paper report error bars suitably and correctly defined, or other appropriate information about the statistical significance of the experiments?
   Answer: [NA]
   Justification: The results are presented as median scores from multiple runs (e.g., 16 runs for the heatmaps) or as direct outputs from single experiments. Error bars, confidence intervals, or other measures of statistical variance are not included in the figures or tables.

8. **Experiments compute resources**
   Question: For each experiment, does the paper provide sufficient information on the

computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
Answer: [Yes]
Justification: The paper provides the execution time of the experiments, as well as the memory constraints and architecture used.

9. **Code of ethics**
Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?
Answer: [Yes]
Justification: The research uses standard, publicly available image datasets for a technical application (image reconstruction) and does not involve human subjects, sensitive data, or applications with immediate negative ethical implications. The work conforms to standard academic research practices.

10. **Broader impacts**
Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
Answer: [NA]
Justification: The paper's conclusion mentions positive applications in feature selection for machine learning models, but does not discuss potential negative societal impacts. The research is foundational and not directly tied to a specific real-world application where such impacts would be immediate.

11. **Safeguards**
Question: Does the paper describe safeguards that have been put in place for the responsible release of data or models that have a high risk for misuse?
Answer: [Yes]
Justification: The research is based on a pre-existing, public dataset (ImageNet), which is cited accordingly.

12. **Licenses for existing assets**
Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited, and are the license and terms of use explicitly mentioned and properly respected?
Answer: [Yes]
Justification: The paper utilizes the ImageNet dataset and cites the original paper that introduced it. The terms of service document is long, and we did not include it in the paper. The terms of service are properly respected.

13. **New assets**
Question: Are new assets introduced in the paper well documented, and is the documentation provided alongside the assets?
Answer: [NA]
Justification: The paper does not introduce or release any new assets, such as datasets, code, or models.

14. **Crowdsourcing and research with human subjects**
Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?
Answer: [NA]
Justification: This research does not involve crowdsourcing or human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**
Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?
Answer: [NA]
Justification: This research does not involve human subjects and therefore does not require IRB approval.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

Answer: [NA]

Justification: The core methodology of this research is based on Self-Organizing Maps (SOMs), a classical neural network algorithm, and does not involve the use of Large Language Models (LLMs).