Generalized Translation Invariant Pattern Classification With Sparse Feature Resonator Networks

Editors: Francisco Acosta, Simone Azeglio, Chase van de Geijn, Sophia Sanborn, Christian Shewmake, Nina Miolane

Abstract

The ability to handle invariant transformations is still an open problem in artificial intelligence. General invariance to transformations like translation are not naturally learned by supervised training, and many network architectures fail with input transformations not covered by the training data. Here, we take an approach based on analysis-by-synthesis, where a generative model describes the construction of simple scenes containing MNIST digits and their transformations. Our approach defines the construction of objects within the scene based on a set of sparse features that are then given an arbitrary translation and color. The resonator network can then be defined to invert the generative model. Sparse features learned from training data act as a basis set to provide flexibility in representing variable shapes of objects. Through an iterative process, the network localizes objects and factors out translation from the sparse features that compose the objects. Objects centered by the resonator network can then be classified using simple logistic regression or deep learning. The classification layer is trained solely on centered data, requiring much less training data, and the network as a whole can identify objects with arbitrary translations. The natural attention-like mechanism of the resonator network also allows for analysis of scenes with multiple objects, where the network dynamics selects and centers only one object at a time.

Keywords: Generative model, Transform invariance, Resonator networks

1. Introduction

Our approach to analysis-by-synthesis (Renner et al., 2024) relies on a specification of a generative model and uses “search in superposition” with a recurrent network architecture called a resonator network (Frady et al., 2020). In this previous work, the input scenes were constructed of rigid fixed objects. Here, we extend the approach to handle objects that have variability in their shapes. Our input scenes are now generated using MNIST digits, which results in variability of each object class. The objects are given an arbitrary translation and coloration.

Following (Frady et al., 2022), the image $Im(x, y)$ is encoded as a function over the pixel space via the superposition of index vectors weighted by their corresponding image pixel values $\mathbf{s} = \sum_{x,y} Im(x, y) \cdot \mathbf{h}^x \odot \mathbf{v}^y$. This image encoding has pivotal properties for enabling scene factorization. Most importantly, it ensures that the *equivariant vector operation* for image translation is the binding operation, i.e. $\mathbf{s} \odot \mathbf{h}^{\Delta x} \odot \mathbf{v}^{\Delta y}$ is the VFA representation of the image translated by $\Delta x, \Delta y$. This vector representation of the input scene can be parsed by a resonator network (Fig. 1A).

The resonator network is a recurrent network that iteratively searches over factor configurations to find an explanation of the input image. An example of its dynamics is

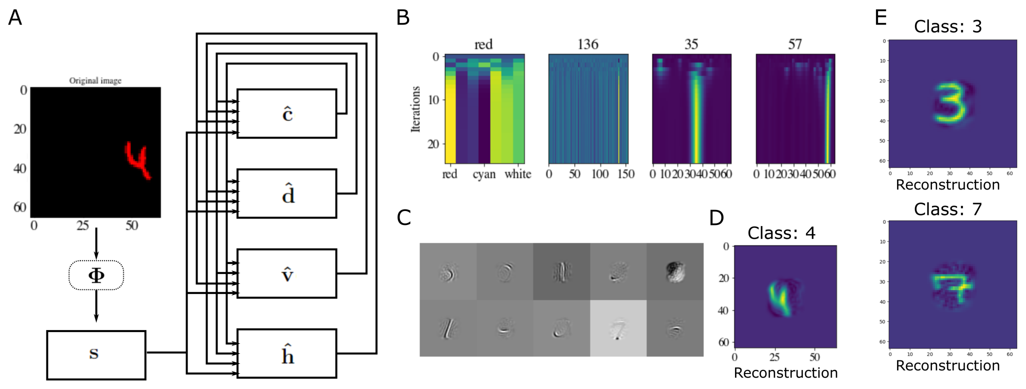

Figure 1: **Sparse Feature Resonator Network. A.** A simple scene with an MNIST digit is presented. The task is to factorize shape, location, and color. **B.** Recurrent resonator network iterates over factorization states, converging from chaos to a stable solution. **C.** Sparse features of MNIST digits are learned from a separate training set (Renner et al., 2024), replacing rigid templates. **D.** The sparse code module reconstructs object shape after factoring out location. **E.** A classifier predicts digit identity from the centered digit; translation invariance is handled by the resonator network.

visualized in Figure 1B. The network iterates for a few timesteps searching over the configuration space, until it finds a good match and rapidly converges to a solution. Once it finds a good solution it remains stable and components describing an object can be interpreted from each module of the resonator network.

In order to handle the natural variability of MNIST digits, we update the generative model to assume that objects in the scene are constructed from a basis set of sparse features, rather than rigid templates. The sparse features are learned from the training set of MNIST digits. In this procedure, we use PCA and ICA to decompose the MNIST digits.

2. Methods and Results

All experiments with the resonator network were implemented in Python using NumPy and PyTorch. We explored two vector dimensions for the benchmarking experiments: N = 10,000 and N = 30,000.

All experiments with the resonator network and different classifiers are done using the test MNIST dataset. The dataset consists of digits that have been size-normalized and centered in a fixed-size 28x28 pixel image. The images were further centered within the 28x28 image by calculating the center of mass of the pixels and translating the image to position this point at the center of the 28x28 field. For the sake of our experiments, these images were placed in the larger 56x56 pixel image.

To evaluate the resonator model, the MNIST dataset is divided into two subsets - training and testing datasets. After the split, the training dataset contains 60,000 images, and the testing dataset contains 10,000 images.

The project's goal is twofold: 1.) to factorize a particular MNIST digit shape from color, and position. And, 2.) to classify correctly factorized MNIST digits. Once an MNIST digit's shape is found after factorization by the resonator network, the corresponding image is reconstructed from sparse feature components. The reconstructed image is cropped to 28x28 image, and scaled back to a grayscale image and fed as an input to a classifier. We used two classifiers: linear classifier and deep neural network(DNN) classifier.

For this task a random scene is generated, 3 digits from the MNIST testing dataset are chosen randomly. The digits are placed in the scene with a random floating point number uniformly distributed between -19 and +19 (to avoid cropping of digits). Then the digits are colored in one of 7 random colors. There are 7 colors used in the generative model given by a matrix $\mathbf{B} \in \mathbb{R}^{3\times7}$ with, for instance, $\mathbf{B}_{cyan} = [0, 1, 1]$. The VSA codebook for colors is $\mathbf{C} = \mathbf{GB}$. The pixels of the scene are then encoded into a vector: $\mathbf{s} = \sum_{x,y,c} Im(x, y, c)h^x \odot v^y \odot Cc$.

The scene is then presented to the resonator network, which iterates in its dynamics until convergence. The full dynamic equations for the resonator network:

$$\begin{aligned}
\hat{\mathbf{c}}(t+1) &= f\left(\acute{\mathbf{C}}\acute{\mathbf{C}}^{\dagger}\left(\mathbf{s}\odot\hat{\mathbf{p}}^{*}(t)\odot\hat{\mathbf{v}}^{*}(t)\odot\hat{\mathbf{h}}^{*}(t)\right)\right),\\
\hat{\mathbf{p}}(t+1) &= f\left(\acute{\mathbf{P}}\acute{\mathbf{P}}^{\dagger}\left(\mathbf{s}\odot\hat{\mathbf{c}}^{*}(t)\odot\hat{\mathbf{v}}^{*}(t)\odot\hat{\mathbf{h}}^{*}(t)\right)\right),\\
\hat{\mathbf{v}}(t+1) &= f\left(\mathbf{V}\mathbf{V}^{\dagger}\left(\mathbf{s}\odot\hat{\mathbf{p}}^{*}(t)\odot\hat{\mathbf{c}}^{*}(t)\odot\hat{\mathbf{h}}^{*}(t)\right)\right),\\
\hat{\mathbf{h}}(t+1) &= f\left(\mathbf{H}\mathbf{H}^{\dagger}\left(\mathbf{s}\odot\hat{\mathbf{p}}^{*}(t)\odot\hat{\mathbf{v}}^{*}(t)\odot\hat{\mathbf{c}}^{*}(t)\right)\right),
\end{aligned} \tag{1}$$

with $f(\mathbf{x})_i = x_i/|x_i|$ (phasor projection) or $f(\mathbf{x})_i = x_i/||\mathbf{x}||_2$ (normalization) and $\mathbf{V}$, $\mathbf{H}$ the codebooks of uncorrelated vectors representing vertical and horizontal coordinates of pixels. A linear transform of the form $\mathbf{V}\mathbf{V}^{\dagger}$ is essentially a linear auto-associative memory that aligns the output to the vectors closest to the input, stored in $\mathbf{V}$.

Summary of classification results on MNIST data sets is shown in table Table 1. The table presents classification accuracies of various models trained and tested on original and reconstructed MNIST data. While the resonator network facotrs out the color and location of the object, the output image is not a perfect reconstruction of the original digit. We trained the classifiers on both the original digits, as well as digits reconstructed by the resonator network. The latter helps the classifier account for reconstruction artifacts due to the factorization procedure, and improves classification performance. We used both a simple Linear Classifier layer as well as LeNet5. The Linear Classifier II, trained and tested on reconstructed data, achieved 76.3%. The LeNet5 trained and tested on reconstructed data reached 80.4%, indicating that end-to-end training on the altered image domain allows the models to partially adapt and recover performance.

3. Discussion

Here, we present a network architecture that can automatically factorize location and color of an object from an object's shape. We have extended previous approaches by allowing for

Table 1: Accuracy of Translation/Color Invariant Classifiers on MNIST scenes

Classifier	Training Type	Accuracy
Linear I	Original	60.2
LeNet5 I	Original	66.7
Linear II	Reconstructed	76.3
LeNet5 II	Reconstructed	80.4

generic objects by using a sparse feature basis, allowing us to factorize previously unseen digit shapes. This then allows us to build simple classifier to have translation invariant classification without any extra training effort. The role of factorization in scene understanding is critical to better extend the notion of invariance to different types of transformations. Further use of such multiplicative operations can lead to more expressive generative models, but then also require more complex factorization procedures.

References

E Paxon Frady, Spencer J Kent, Bruno A Olshausen, and Friedrich T Sommer. Resonator networks, 1: An efficient solution for factoring high-dimensional, distributed representations of data structures. *Neural Computation*, pages 1–21, oct 2020.

E Paxon Frady, Denis Kleyko, Christopher J Kymn, Bruno A Olshausen, and Friedrich T Sommer. Computing on functions using randomized vector representations (in brief). In *Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference*, pages 115–122, 2022.

Alpha Renner, Lazar Supic, Andreea Danielescu, Giacomo Indiveri, Bruno A Olshausen, Yulia Sandamirskaya, Friedrich T Sommer, and E Paxon Frady. Neuromorphic visual scene understanding with resonator networks. *Nature Machine Intelligence*, 6(6):641–652, 2024.