Convolutional Neural Generative Coding: Scaling Predictive Coding to Natural Images

Alexander G. Ororbia (ago@cs.rit.edu)
Rochester Institute of Technology, 1 Lomb Memorial Dr
Rochester, NY 14620 USA

Ankur Mali (ankurarjunmali@usf.edu)
University of South Florida, 4202 E Fowler Ave
Tampa, FL 33620 USA

Abstract

In this work, we develop *convolutional neural generative coding* (Conv-NGC), a generalization of predictive coding to the case of convolution/deconvolution-based computation. Specifically, we concretely implement a flexible neurobiologically-motivated algorithm that progressively refines latent state feature maps in order to dynamically form a more accurate internal representation/reconstruction model of natural images. The performance of the resulting sensory processing system is evaluated on complex datasets such as Color-MNIST, CIFAR-10, and SVHN. We study the effectiveness of our brain-inspired model on the tasks of reconstruction and image denoising and find that it is competitive with convolutional auto-encoding systems trained by backpropagation of errors and outperforms them with respect to out-of-distribution reconstruction (including the full 90k CINIC-10 test set).

Keywords: Predictive coding; Brain-inspired learning; Computer vision, neuromorphic hardware, convolution

Introduction

The algorithm known as backpropagation of errors (Werbos, 1981; Linnainmaa, 1970) (or backprop) has served as a crucial element behind the tremendous progress that has been made in recent machine learning research, progress which has been accelerated by advances made in computational hardware as well as the increasing availability of vast quantities of data. Nevertheless, despite reaching or surpassing human-level performance on many different tasks ranging from those in computer vision (He, Zhang, Ren, & Sun, 2015) to game-playing (Silver et al., 2016) to generative modeling (Rombach, Blattmann, Lorenz, Esser, & Ommer, 2022), the field still has a long way to go towards developing artificial general intelligence. In order to increase task-level performance, the size of deep networks has increased greatly over the years, up to hundreds of billions of synaptic parameters as seen in modern-day transformer networks (Floridi & Chiriatti, 2020). However, this trend has started to raise concerns related to energy consumption (Patterson et al., 2021) and as to whether such large systems can attain the flexible, generalization ability of the human brain (Brown et al., 2020). Furthermore, backprop itself imposes additional limitations beyond its long-argued biological implausibility (Crick, 1989; Gardner, 1993; Shepherd, 1990), such as its dependence on a global error feedback pathway for determining each neuron's individual contribution to a deep network's overall performance resulting in sequential backward, non-local updates that make parallelization difficult (which stands in strong contrast to how learning occurs in the brain (Jaderberg et al., 2016; A. G. Ororbia, Mali, Kifer, & Giles, 2018; A. G. Ororbia & Mali, 2019)). The limitations imposed by the prohibitively large size of these systems as well as the constraints imposed by their workhorse training algorithm, backprop, have motivated the investigation and development of alternative methodology.

Some of the most promising pathways come from the emerging domain of research known as brain-inspired computation, which seeks to develop neural architectures and their respective credit assignment algorithms that leverage only local information, motivated strongly by how learning is conducted by the brain. The promise of brain-inspired computing brings with it synaptic adjustment mechanisms that are neurobiologically-grounded (Hebb, 1949) as well as neural computation and inference that is flexible, capable of conducting a wide variety of operations (A. Ororbia & Kifer, 2022) at biologically more faithful levels (Maass, 1997; A. Ororbia, 2019), facilitating massive algorithmic parallelization (at scale) and adaption on analog and neuromorphic hardware (Furber, 2016; Roy, Jaiswal, & Panda, 2019; Kendall, Pantone, Manickavasagam, Bengio, & Scellier, 2020).

In addressing the challenges facing backprop-based ANNs and in the direction of brain-inspired computing, we design a new model for image processing, convolutional neural generative coding (Conv-NGC), which is inspired by human learning. Human information processing is often interpreted as an interaction of hierarchical feedforward and feedback (backward) projections that continuously *predict and correct* internal neural representations of that information (Rao & Ballard, 1999). In computational neuroscience, this interplay is known as predictive coding (Rao & Ballard, 1999; Rainer, Rao, & Miller, 1999). Similarly, Conv-NGC includes state prediction and correction steps that continuously generate and refine its internal representations (Figure 2). Furthermore, Conv-NGC encodes complex visual information by incorporating blocks of (de)convolution into a top-down directed generative model within the framework of predictive coding (Bastos et al., 2012; Chalasani & Principe, 2015; Clark, 2015; A. Ororbia, Mali, Giles, & Kifer, 2022; A. Ororbia & Kifer, 2022).

Our contributions are as follows: **1)** we propose a new neural perception model, Conv-NGC, which acquires robust representations of natural images in an unsupervised fashion[1], **2)** to

[1]Code can be found at: https://github.com/ago109/conv-pc

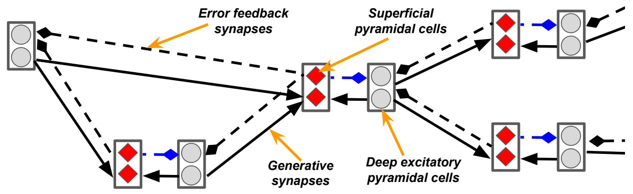


Figure 1: An arbitrary NGC circuit. Grey circles are state units (deep excitatory pyramidal cells) and red diamonds are error units (superficial pyramidal cells). Solid arrows are predictive synapses and dashed diamond ones are error synapses.

our knowledge, this is the first work in the literature where visual inputs are processed using deep (de)convolutional layers naturally within of framework of predictive coding, significantly enhancing its representation power for vision-based tasks, and, **3)** we demonstrate, on natural image datasets, that the proposed Conv-NGC is competitive with existing backprop-based models (of similar architectural designs) on the tasks of image reconstruction and denoising and outperforms them with respect to out-of-distribution prediction.

Convolutional Neural Generative Coding

We start by describing our model instantiation of convolutional neural generative coding (Conv-NGC), which is tasked with learning from streams of natural images in an unsupervised fashion. Typically, in a visual recognition task, input patterns belonging to different object classes, arranged into batches, are presented to a processing system at different time points for training. In this section, we describe our problem setup and model architecture (see Appendix[2] for related work, relationships with Conv-NGC, and neurobiological motivations).

Problem Setup: With respect to the problem setup, a Conv-NGC system is tasked with processing a finite collection of images depicting a certain set of object classes arranged in an arbitrary order. The dataset contains a set of n input samples: $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1 \dots (\mathbf{x}_n, \mathbf{y}_n)\}$. Here, $\mathbf{x}_j \in \mathcal{R}^{I \times I \times C}$ represents the image of the j^{th} input sample ($I \times I$ is the 2D shape of any single image channel and C is the number of channels, i.e., $C = 3$ for the red-green-blue channels) and $\mathbf{y}_j \in \{0,1\}^{Y \times 1}$ is its ground truth class label (Y is the number of classes). Note that while we formalize the labels available in each benchmark dataset, the models that we study in this work are unsupervised and, as a result, never make use of the labels in $\mathcal{D}$.

Notation: In this study, the symbol $*_s$ is used to refer to a strided convolution where s is the stride argument ($*_1$ means convolution with stride of 1, which would also be the same as just $*$). In contrast, the symbol $\circlearrowleft_s$ denotes deconvolution (or transposed convolution) with a stride of s. The Hadamard product is denoted by $\odot$ while $\cdot$ represents a matrix/vector multiplication. $()^T$ denotes the transpose operation. Flatten($\mathbf{z}$) means that the input tensor $\mathbf{z}$ is converted to a column vector with a number of rows equal to the number of elements that it originally contained while UnFlatten($\mathbf{z}$) is its inverse (i.e., it converts the vector back to its original tensor shape). Finally, Dilate($\mathbf{v}, s$) is used to represent a dilation function controlled by the dilation (integer) size (e.g., $s = 2$). Note that Conv-NGC is technically made up of 4D synaptic tensors, thus, when we write $\mathbf{W}_{ij}$, we are saying that we are retrieving a 2D matrix at position (i, j) in the 4D tensor $\mathbf{W}$ (for extracting a scalar in $\mathbf{W}$, one would write W_{ijkl}, without bold font).

Deep Convolutional Neural Coding

Conv-NGC is a generalization of the NGC computational framework in (A. Ororbia & Kifer, 2022) to the case of natural image data. The underlying process of Conv-NGC can be divided into three components: **1)** local prediction and error unit map calculation, **2)** latent state map correction, and **3)** local synaptic adjustment. See Figure 1 for a depiction of a general NGC circuit (and see Figure 2 for Conv-NGC).

The Neural Coding Process: Fundamentally, Conv-NGC consists of a set of L predictive layers (typically arranged hierarchically, though this is not a strict architectural requirement, e.g., Figure 1 depicts a non-hierarchical NGC circuit) that are designed to learn latent representations of observed patterns. It is important to note that, unlike the bottom-up forward propagation of a standard convolutional neural network (CNN), neural layers within a Conv-NGC system make top-down predictions, the errors of which are then used to subsequently correct the layers' own values. In effect, this means that the layers in Conv-NGC are stateful and their computation within a forward pass can be further broken down into distinct computations – top-down prediction and state correction.

First, in the top-down prediction phase, given its current state $\mathbf{z}^\ell$ (which abstractly models the functionality of deep excitatory pyramidal cells), each layer ℓ of our model tries to predict the state of the layer below it, yielding prediction $\bar{\mathbf{z}}^{\ell-1}$. At the bottom-most layer, the model predicts $\bar{\mathbf{z}}^0_x$ for the input data pattern ($\mathbf{x}$). Following this, a set of error neurons (which abstractly model the functionality of superficial pyramidal cells) compute the mismatch between this prediction and the actual state $\mathbf{z}^{\ell-1}$, i.e., $\mathbf{e}^{\ell-1} = (\mathbf{z}^{\ell-1} - \bar{\mathbf{z}}^{\ell-1})$.

Second, in the correction phase, the model's internal state layers are corrected based on how accurate their top-down guesses turned out to be. The error/mismatch signal $\mathbf{e}^{\ell-1}$ computed during the prediction phase is subsequently used to adjust the current values of the state $\mathbf{z}^\ell$ that originally made the prediction $\bar{\mathbf{z}}^{\ell-1}$. This *local error correction*, which is not present in traditional feedforward ANNs, helps to nudge the state $\mathbf{z}^\ell$ towards a configuration (i.e., set of values) that better predicts the layer below in the future and thus moves the layer towards a better representation/higher-level abstraction of the input. This correction is "local" in the sense that each layer's update depends only on a top-down error signal, which is produced by comparing its own values with the predictions made by the layer above it, and a bottom-up error signal, which is produced by comparing its predictions of a nearby layer's activity values and that state's current actual values.

Given the description of the two general computations above, we may now describe how Conv-NGC processes data.

[2]**Appendix can be found at:** https://shorturl.at/hMN89

Unlike a standard feedforward ANN, which predicts $\mathbf{y}_j$ given $\mathbf{x}_j$ (or, in the case of auto-encoding, the ANN attempts to reconstruct $\mathbf{x}_j$ itself given $\mathbf{x}_j$ as input) with a single forward pass, our model works in multiple steps. First, a Conv-NGC network predicts the value of $\mathbf{z}^0 = \mathbf{x}_j$ from $\mathbf{z}^1$ (which generates prediction $\bar{\mathbf{z}}^0$), $\mathbf{z}^1$ from $\mathbf{z}^2$ (which generates prediction $\bar{\mathbf{z}}^1$), etc. As each prediction is made, the corresponding set of error neurons compute the mismatch between the predicted value and its target state layer.[3] Next, the neural system then corrects the values of its states $\{\mathbf{z}^1, \mathbf{z}^2, \cdots, \mathbf{z}^L\}$ given the current values of the error neurons. These two steps are then repeated for several iterations, i.e., over a stimulus window of T steps, to arrive at a set of internal representations that accurately represent the input. This means that each layer/region ℓ of any NGC circuit tries to satisfy two main objectives: 1) to uncover a better latent representation in order to predict a nearby neural region/layer (a bottom-up adjustment), and 2) to be closer to what the layer above ($\mathbf{z}^{l+1}$) predicted its state should be (a top-down expectation). By performing several steps of top-down prediction and state correction, the model minimizes layer-specific predictions while optimizing its global representation for the current dataset. Implementation details of the above processes, specifically generalized to the case of feature maps and (de)convolution, are provided in the next section.

Neural Coding Training and Inference

In this section, we provide concrete implementation details of the neural coding process described earlier, depicting how layer-wise state prediction, state-correction, and synaptic parameter updating occur specifically in the context of visual object reconstruction. The first two steps iteratively predict and correct the representations of the Conv-NGC model for observed input values (natural images) of the current dataset. After T iterations, the final step entails adjusting model synaptic efficacies using simple Hebbian-like updates. We start by providing the mechanics of the three above steps and then describe the objective function that our model dynamically optimizes. The full algorithmic specification of Conv-NGC is presented in the pseudocode in the Appendix and an example 3-layer Conv-NGC model is visually depicted in Figure 2.

Inference: Predicting and Correcting Neural States

For each layer of a Conv-NGC system, note that the full state of any given layer ℓ is represented by a set of C_ℓ feature state maps, i.e., $\{\mathbf{z}^\ell_1, \mathbf{z}^\ell_2, ..., \mathbf{z}^\ell_{C_\ell}\}$ where each state map $\mathbf{z}^\ell_i$ is essentially a block/cluster of neurons (that encode a partial distributed representation of the detected input below), meaning that any layer consists of C_ℓ channels. To initialize each state map, instead of setting each to be a grid of zero values (similar to how neural vectors are initialized in fully-connected predictive coding models (A. Ororbia & Kifer, 2022)), we initialize the states with a top-down ancestral projection pass by first sampling a random noise value for the top-most set of latent state maps, i.e., $\mathbf{z}^L_c \sim \mathcal{N}(\mu_z, \sigma_z)$, $c = 1, ..., C_L$ (we set $\mu_z = 5$ and $\sigma_z = 0.05$ in this work), and then project this sampled state down along the Conv-NGC network to obtain the initial values of the other layers (see Appendix for details).

Layer-wise State Map Prediction: At each layer, the ith feature map state $\mathbf{z}^\ell_i$ is used to (partially) predict the jth feature map state of the layer below it, producing the prediction $\bar{\mathbf{z}}^{\ell-1}_j$. Note that a vector representation of a layer's entire state is the concatenation of all of its C (flattened) feature maps, i.e., $\mathbf{z}^\ell = [\mathbf{z}^\ell_1, \mathbf{z}^\ell_2, \cdots, \mathbf{z}^\ell_C]$. These local layer-wise predictors perform their computation independently (in parallel) and are coordinated through error units at each layer ℓ for each feature map. Specifically, the error neurons $\mathbf{e}^{\ell-1}_j$ at layer $\ell-1$ (for the jth channel) compute the difference between the prediction $\bar{\mathbf{z}}^{\ell-1}_j$ (from the layer ℓ above) and the target map activity $\mathbf{z}^{\ell-1}_j$. This error message, in turn, is used to (partially) adjust the state representation $\mathbf{z}^\ell$ at layer ℓ. Formally, the predictor and error neurons are computed on a per-feature map basis as follows:

$$\bar{\mathbf{z}}^{\ell-1}_j = g^\ell\Big(\Big(\sum_i^{C_\ell} \mathbf{W}^\ell_{ij} \circlearrowleft_s \phi^\ell(\mathbf{z}^\ell_i)\Big) + \mathbf{b}^\ell_j\Big),\ \mathbf{e}^{\ell-1}_j = (\mathbf{z}^{\ell-1}_j - \bar{\mathbf{z}}^{\ell-1}_j), \tag{1}$$

where $\phi^\ell()$ is the nonlinear activation activation applied to the state activity map $\mathbf{z}^\ell_i$, $\mathbf{W}^\ell_{ij}$ is the jth learnable kernel for the ith input state map, and $\mathbf{b}_j$ is the jth bias map applied to the prediction/output map $\bar{\mathbf{z}}^\ell_j$. $g^\ell()$ is the predictive output nonlinearity applied to the result $\bar{\mathbf{z}}^\ell_j$ (chosen according to the distribution that is chosen to model the state of $\ell-1$, e.g., the identity for a multivariate Gaussian model). Notice in Equation 1, for Conv-NGC, that what would be a scalar state value in the models of (Rao & Ballard, 1999; A. Ororbia & Kifer, 2022)) is now a 2D matrix (channel) of state activities that gets expanded through the application of kernel parameters $\mathbf{W}^\ell_{ij}$ with stride s (each predictive synapse of (Rao & Ballard, 1999; A. Ororbia & Kifer, 2022)) is replaced with a kernel).

Note that in the above equation, in a (de)convolutional layer, a final complete prediction of the jth output channel involves the summation/aggregation of multiple filters applied to each (input) state feature map of layer ℓ (up to C_ℓ input channels). Extending the prediction operation of Figure 1 to make use of stacks of operations, such as within a residual block, is detailed in the Appendix (which contains a biologically-plausible scheme for handling credit assignment for operation blocks).

State Map Correction: During this step, the (i-th) state feature map $\mathbf{z}^\ell_i$ is refined using the values of the error neurons of the current and previous layers, i.e., ℓ and $\ell-1$. For the top-most layer, the error $\mathbf{e}^L_i$ of any feature map i does not exist so it is not computationally modeled/simulated. Formally, the ℓ-th predictor (or rather, its (i-th) map $\mathbf{z}^\ell_i$) corrects its state values using both the bottom-up and top-down error messages/signals according to the following (Euler) integration scheme below:

$$\mathbf{z}^\ell_i \leftarrow \big(\mathbf{z}^\ell_i + \beta\mathbf{d}^\ell_i - \gamma\mathbf{z}^\ell_i\big), \text{where } \mathbf{d}^\ell_i = -\mathbf{e}^\ell_i + \sum_j^{C_{\ell-1}} \mathbf{E}^\ell_{ji} *_s \mathbf{e}^{\ell-1}_j, \tag{2}$$

[3]These predictions and error neuron calculations, although presented as occurring sequentially, can naturally be made in parallel given that they depend on immediately/locally available values.

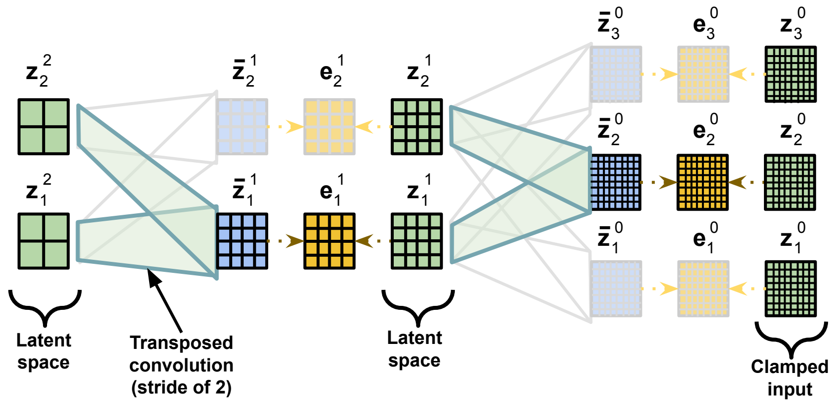


Figure 2: A 3-layer convolutional neural generative coding network. Each latent state map $\mathbf{z}_i^\ell$ generates a prediction of each latent map $\mathbf{z}_j^{\ell-1}$ in the layer below via deconvolution. In non-transparent colors, a possible prediction pathway within the circuit is shown (transparent colors indicate no involvement in this pathway), i.e., state maps $\mathbf{z}_1^2$ and $\mathbf{z}_2^2$ each contribute to the prediction $\bar{\mathbf{z}}_1^1$ of map $\mathbf{z}_1^1$ while state maps $\mathbf{z}_1^1$ and $\mathbf{z}_2^1$ each contribute to the prediction $\bar{\mathbf{z}}_2^0$ of map $\mathbf{z}_2^0$ (note: in the bottom layer, each $\mathbf{z}_j^0$ map could represent a particular color channel of an input image). Error unit map $\mathbf{e}_1^1$ encodes the mismatch between prediction $\bar{\mathbf{z}}_1^1$ and the value of latent map $\mathbf{z}_1^1$ while error unit map $\mathbf{e}_2^0$ encodes the mismatch between prediction $\bar{\mathbf{z}}_2^0$ and the value of latent map $\mathbf{z}_2^0$. Green squares depict latent unit maps, blue squares depict prediction output maps, and orange/yellow squares depict error unit maps.

where $\leftarrow$ denotes a variable override and the modulation/adjustment factor β is a constant value, which controls the state correction rate, and $-\gamma \mathbf{z}_i^\ell$ is the leak variable, controlled by the strength factor γ (set to a small value such as 0.001, also serving as a form of light regularization applied to the latent state map neurons). The (4D) error kernel tensor $\mathbf{E}^\ell$ models a learnable feedback pathway that is responsible for transmitting the error from the layer $\ell - 1$ to the layer ℓ. This means that the bottom-up error message/perturbation is produced by aggregating across the $C_{\ell-1}$ output channels/state maps that make up layer $\ell - 1$, i.e., a convolution must be applied to each output channel j as follows: $\mathbf{E}_{ji}^\ell *_s \mathbf{e}_j^{\ell-1}$, meaning that the prediction of each lower-level state map that was (partially) made by the state map/predictor $\mathbf{z}_i^\ell$ contributes equally to its final value, resulting in a state map correction (which is further modulated by the top-down pressure $\mathbf{e}_i^\ell$, exerted by the layer above $\mathbf{z}^{\ell+1}$). Although having separate error kernels for transmitting error message signals resolves the weight transport problem that characterizes backprop-based learning (and is thus more biologically plausible), this comes at an increased memory footprint and additional computation which is needed for updating the error filters themselves. Thus, we set the error filters, in this work, to be $\mathbf{E}_{ji}^\ell = (\mathbf{W}_{ij}^\ell)^T$, to speed up simulation (not observing any noticeable change in performance). Note that, for test time inference, we do not adjust synaptic efficacies and either clamp or initialize the bottom state $\mathbf{z}^0 = \mathbf{x}$. To obtain a prediction, the neural system will conduct T steps of state prediction and correction, eventually outputting $\bar{\mathbf{z}}^0$.

Training: Updating Model Parameters

Neural Coding Synaptic Update: After T iterations of state prediction and correction (as in the previous section), the updates to each state prediction filter $\mathbf{W}_{ij}^\ell$ and error filter $\mathbf{E}_{ji}^\ell$ are computed according to a Hebbian-like update:

$$\Delta \mathbf{W}^\ell{}_{ij} = \mathbf{e}_j^{\ell-1} *_1 \text{Dilate}\Big(\big(\phi^\ell(\mathbf{z}_i^\ell)\big)^T, s\Big),$$
$$\Delta \mathbf{E}^\ell{}_{ji} = \lambda\Big(\text{Dilate}\Big(\big(\phi^\ell(\mathbf{z}_i^\ell)\big)^T, s\Big) *_1 \mathbf{e}_j^{\ell-1}\Big)$$

where λ is modulation factor meant to control the time scale of the evolution of the error filters (and generally set to < 1.0, e.g., 0.9) - note that this part of the update rule is discarded if $\mathbf{E}_{ji}^\ell = (\mathbf{W}_{ij}^\ell)^T$ (which can be done to save space and minimal change in performance). The above local update rule is a generalization of the one proposed in (A. Ororbia et al., 2022; A. Ororbia & Kifer, 2022) to the case of a (de)convolutional filter. After the update for a particular filter has been calculated and it has been used to adjust the current physical state of the kernel synapses, we further normalize/constrain each kernel such that its Euclidean norm does not exceed one (see the Appendix for the specification of the re-projection step). This constraint ensures that Conv-NGC avoids the degenerate solution of simply increasing its synaptic kernel values while obtaining only small/near-zero latent activity values, much as is done in convolutional sparse coding (Heide, Heidrich, & Wetzstein, 2015) (this also means that one could also view Conv-NGC as a sort of "deep" convolutional sparse coding).

Objective Function: During training, a Conv-NGC model refines its internal states such that the output of the local predictions move as close as possible to the actual values of the state maps, which means that the bottom-most layer stays as close as possible to the sensory input $\mathbf{x}_j$. In order to do this, Conv-NGC optimizes *total discrepancy optimization* (ToD) (A. G. Ororbia, Haffner, Reitter, & Giles, 2017; A. G. Ororbia & Mali, 2019; A. Ororbia & Kifer, 2022) via the prediction, state correction, and synaptic adjustment steps presented in the earlier sub-sections. The simplest version of this optimization function is defined as the sum of mismatches between predictions and actual states at each level of the model:

$$\mathcal{L}_{ToD} = \sum_{\ell} \Big(-\frac{1}{2} ||\mathbf{z}^{\ell} - \bar{\mathbf{z}}^{\ell}||_2^2 \Big) \tag{3}$$

The online minimization of the discrepancy among states as depicted in Figure 3 progressively refines the representation of all layers in our model. Notably, the total discrepancy objective can also be viewed as approximately minimizing a form of (variational) free energy (Friston, 2010), representing a concrete statistical learning connection to and implementation of a prominent neuro-mechanistic Bayesian brain theory.

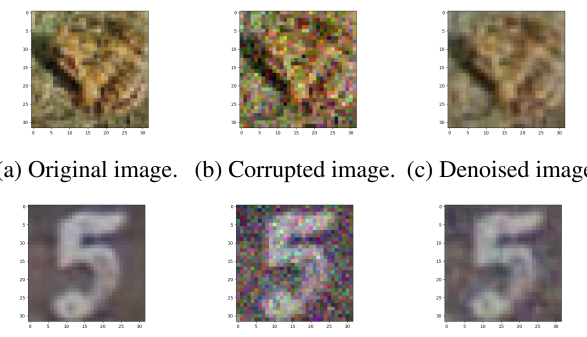
Experiments

Given our specification of the Conv-NGC model's inference and learning processes, we next describe the experimental setup. We compare Conv-NGC to powerful backprop-based models such as the convolutional autoencoder (Conv-AE) and the denoising convolutional autoencoder (Conv-DAE) as well as the fully-connected form of neural generative coding (NGC-PCN), which under certain conditions, recovers the unsupervised form of the model in (Salvatori et al., 2021) (the hierarchical predictive coding network, or PCN). The NGC-PCN represents our base comparison to standard predictive coding.

With respect to the datasets used in our simulations, we utilized Color-MNIST, CIFAR-10, and SVHN (the Street View House Numbers database) to both train and evaluate the baselines and Conv-NGC (using respective training/validation/testing splits) while CINIC-10 was only used as an additional test set for the out-of-distribution reconstruction experiments. All datasets consisted of 32×32 complex natural images. In the Appendix, we provide baseline model and Conv-NGC configuration descriptions and training details, hyperparameter settings, and dataset details.

Reconstruction and Denoising: In this set of tasks, we investigate each model's ability to reconstruct the given input images from each dataset as well as to recover the original image values under corruption noise. For the task of reconstruction, we measure the mean squared error (MSE) for each input between the model's predicted values $\hat{\mathbf{x}}_i$ and the original image $\mathbf{x}_i$. To better probe the image quality of reconstructed outputs, we evaluate the structural similarity index measure (SSIM). Finally, for the more complex natural images in SVHN and CIFAR-10, we measure peak signal-to-noise ratio (PSNR). Note that, in the Appendix, we present the mathematical formulas and details for each of these metrics. For the results of the reconstruction and denoising tasks, see Tables 1 and 2a.

Out-of-Distribution Reconstruction: One interesting property of auto-encoding systems that we are interested in examining is their ability to reconstruct image pattern samples not seen in the original data distribution, particularly samples that come from a much different distribution (violating the typical i.i.d. assumption in statistical learning). To evaluate out-of-distribution (OOD) reconstruction ability, we take the



(a) Original image. (b) Corrupted image. (c) Denoised image.

(d) Original image. (e) Corrupted image. (f) Denoised image.

Figure 3: Example image randomly sampled from a dataset test set (Left), the same image corrupted with noise $\sim \mathcal{N}(0, 0.1)$ (Middle), and the Conv-NGC denoising of the corrupted pattern (Right). Top row shows a sample taken from CIFAR-10 while the bottom row shows one taken from SVHN.

models trained on the SVHN dataset (the source dataset) and evaluate their ability to reconstruct samples from the distinct CIFAR-10 dataset as well as the larger, more difficult CINIC-10 database (specifically evaluating predictive ability on these test sets). For the results of this OOD task, see Table 2b.

Discussion: As seen in our results, Conv-NGC outperforms NGC-PCN (or PCN), as expected, and exhibits competitive behavior with the backprop-based autoencoder models. Notably, in terms of reconstruction, Conv-NGC even offers improved SSIM and PSNR, which is likely the result of its ability to learn a reconstruction process (over a T-length window of time). With respect to image denoising, we observe that the Conv-NGC models works well, outperforming both the AE and NGC models specifically on the harder natural image datasets (CIFAR-10 and SVHN), but under-performs the DAE. The DAE, however, was trained directly for the task of denoising (with noise injected to its input nodes), so it makes sense that it would outperform models that were not tuned to the task (and thus serves well as a soft upper bound on performance). See Figure 3 for visual examples of image samples that the Conv-NGC model denoised (for CIFAR-10 and SVHN).

Note that, for the table metrics reported, pixel reconstruction/model output values were re-scaled to be between $[0, 255]$, which better highlighted the gap between results obtained for Conv-NGC and other baseline models. More importantly, Conv-NGC exhibits a low variance and better visual reconstruction comparatively as indicated by SSIM, which is a metric that more closely correlates with human perception.

To probe the knowledge acquired in the Conv-NGC's distributed representations, we examined the learned latent state feature maps of trained models on sampled test images, taken from each of the data benchmarks. Qualitatively, we observed that Conv-NGC appears to learn a (noisy) form of the image pyramid (Adelson, Anderson, Bergen, Burt, & Ogden, 1984) within its latent activities (see the Appendix for visual samples of this result). Notably, in Conv-NGC, the sensory image

Table 1: Model reconstruction and denoising performance on the test samples of Color-MNIST (top), CIFAR-10 (middle), and SVHN (bottom). Reported measurements are mean and standard deviation across five trials. (Note: model output values are re-scaled to between $[0,255]$ when calculating metrics.) In terms of performance, a lower MSE and higher SSIM are better.

Color-MNIST	Reconstruction		Denoising ($\sim \mathcal{N}(0,0.1)$)	
Model	**MSE**	**SSIM**	**MSE**	**SSIM**
Conv-AE	32.99 ± 1.0680	0.9021 ± 0.0100	176.9321 ± 2.0923	0.6912 ± 0.007
Conv-DAE	23.09 ± 0.4722	0.9324 ± 0.0060	$\mathbf{76.8610 \pm 0.8511}$	$\mathbf{0.8424 \pm 0.003}$
NGC-PCN	328.5659 ± 3.5000	0.7652 ± 0.0032	887.205 ± 18.2500	0.5476 ± 0.0075
Conv-NGC	$\mathbf{11.2802 \pm 0.5008}$	$\mathbf{0.9838 \pm 0.0005}$	202.2222 ± 1.0511	0.6732 ± 0.0006
CIFAR-10	Reconstruction		Denoising ($\sim \mathcal{N}(0,0.1)$)	
Model	**MSE**	**SSIM**	**MSE**	**SSIM**
Conv-AE	13.6890 ± 1.6320	0.9310 ± 0.0030	208.4608 ± 9.3928	0.7628 ± 0.0003
Conv-DAE	8.8810 ± 1.5311	0.9720 ± 0.0010	$\mathbf{16.7781 \pm 5.1037}$	$\mathbf{0.9110 \pm 0.0020}$
NGC-PCN	413.5140 ± 7.1000	0.7230 ± 0.0003	913.2587 ± 21.2000	0.5308 ± 0.0025
Conv-NGC	$\mathbf{6.3668 \pm 1.2522}$	$\mathbf{0.9955 \pm 0.0009}$	183.8624 ± 9.4002	0.8603 ± 0.0007
SVHN	Reconstruction		Denoising ($\sim \mathcal{N}(0,0.1)$)	
Model	**MSE**	**SSIM**	**MSE**	**SSIM**
Conv-AE	7.5043 ± 2.9351	0.8934 ± 0.0002	154.6783 ± 25.7555	0.7490 ± 0.0030
Conv-DAE	2.7263 ± 1.9286	0.9510 ± 0.0002	$\mathbf{87.9926 \pm 7.2091}$	$\mathbf{0.9002 \pm 0.0030}$
NGC-PCN	67.393 ± 9.02	0.9436 ± 0.0028	1116.764 ± 283.55	0.6704 ± 0.0904
Conv-NGC	$\mathbf{1.9500 \pm 0.5609}$	$\mathbf{0.9976 \pm 0.0002}$	97.2695 ± 3.5245	0.8650 ± 0.0007

Table 2: Both in (a) and (b), mean and standard deviation of SSIM, PSNR, and MSE are reported (over five experimental trials). Lower MSE and higher SSIM and PSNR are better.

CIFAR-10	Reconstruction	Denoising ($\sim \mathcal{N}(0,0.1)$)
Model	**PSNR**	**PSNR**
Conv-AE	36.7872 ± 1.0056	24.6527 ± 0.2001
Conv-DAE	38.6462 ± 1.2033	$\mathbf{36.4346 \pm 0.2021}$
NGC-PCN	22.8462 ± 0.0400	19.0410 ± 0.1203
Conv-NGC	$\mathbf{41.0912 \pm 0.0234}$	25.9004 ± 0.2004
SVHN	Reconstruction	Denoising ($\sim \mathcal{N}(0,0.1)$)
Model	**PSNR**	**PSNR**
Conv-AE	39.3777 ± 0.2000	26.2369 ± 0.3000
Conv-DAE	43.7750 ± 0.3000	$\mathbf{28.6864 \pm 0.2000}$
NGC-PCN	33.2763 ± 1.0800	19.8709 ± 2.0800
Conv-NGC	$\mathbf{45.2305 \pm 0.0800}$	28.2513 ± 0.1600

(a) Analysis of model reconstruction and denoising ability

	SVHN to CIFAR-10		
Model	**SSIM**	**PSNR**	**MSE**
Conv-AE	0.69 ± 0.0010	24.03 ± 0.26	387.02 ± 7.82
Conv-DAE	0.77 ± 0.0002	29.46 ± 0.37	176.00 ± 5.09
Conv-NGC	$\mathbf{0.98 \pm 0.0006}$	$\mathbf{35.03 \pm 0.30}$	$\mathbf{26.80 \pm 2.00}$
	SVHN to CINIC-10		
Model	**SSIM**	**PSNR**	**MSE**
Conv-AE	0.70 ± 0.0050	23.74 ± 0.31	275.09 ± 8.02
Conv-DAE	0.79 ± 0.0020	27.10 ± 0.50	127.90 ± 4.55
Conv-NGC	$\mathbf{0.97 \pm 0.0013}$	$\mathbf{32.54 \pm 0.20}$	$\mathbf{47.83 \pm 2.50}$

(b) Out-of-distribution reconstruction performance results.

appears in some of the internal feature state maps, but is a down-sampled, decreased resolution form of itself (much akin to the repeated process of smoothing/sub-sampling that results in lower spatial density images the higher up one goes within the image pyramid). This result complements early work that highlighted the notion that the brain processes visual information at different resolutions (Campbell & Robson, 1968). This biological multi-resolution analysis of the world allows the brain to extract useful information from complex input patterns and it appears that Conv-NGC implicitly learns to conduct a similar type of analysis on the patterns contained in the natural image benchmarks.

Surprisingly, with respect to the OOD experiments, Conv-NGC outperforms all of the baseline models (including the Conv-DAE), further corroborating the in-dataset reconstruction result(s) that Conv-NGC appears to (meta-)learn a type of reconstruction process that works well on unseen, out-of-distribution natural image patterns. This desirable improvement is indicated by the higher trial-averaged (OOD) PSNR as well as the (OOD) SSIM score on both problem settings: 1) SVHN-to-CIFAR-10 reconstruction, and 2) SVHN-to-CINIC-10 reconstruction. What is most impressive is that the Conv-NGC model was trained on natural images (in SVHN) that did not contain any of the objects found within either CIFAR-10 and CINIC-10 and it was still able to reconstruct with top performance. This result offers a promising future direction worth exploring for Conv-NGC/NGC predictive coding systems in general – their ability to generalize to unseen patterns that violate the assumption that they were generated by a distribution similar to that of the training data.

Conclusion

In this work, we proposed convolutional neural generative coding (Conv-NGC), a generalization of a computational predictive coding framework to the case of natural images. Our experiments on three benchmark datasets, i.e., Color-MNIST, CIFAR-10, and SVHN, demonstrate that Conv-NGC is competitive with powerful backprop-based convolutional autoencoding models with respect to both pattern reconstruction and image denoising and notably outperforms all of them in the context of out-of-distribution reconstruction. Our results mark an important step towards crafting more robust, general brain-inspired neural architectures and learning processes capable of handling complex machine learning tasks.

References

Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., & Ogden, J. M. (1984). Pyramid methods in image processing. *RCA engineer*, *29*(6), 33–41.

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., & Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, *76*(4), 695–711.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Askell, e. a., Amanda (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Campbell, F. W., & Robson, J. G. (1968). Application of fourier analysis to the visibility of gratings. *The Journal of physiology*, *197*(3), 551.

Chalasani, R., & Principe, J. C. (2015, Sep.). Context dependent encoding using convolutional dynamic networks. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(9), 1992-2004. doi: 10.1109/TNNLS.2014.2360060

Clark, A. (2015). *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, *337*(6203), 129–132.

Floridi, L., & Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, *30*(4), 681–694.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, *11*(2), 127–138.

Furber, S. (2016). Large-scale neuromorphic computing systems. *Journal of neural engineering*, *13*(5), 051001.

Gardner, D. (1993). *The neurobiology of neural networks*. MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the ieee international conference on computer vision* (pp. 1026–1034).

Hebb, e. a., Donald O. (1949). *The organization of behavior*. New York: Wiley.

Heide, F., Heidrich, W., & Wetzstein, G. (2015). Fast and flexible convolutional sparse coding. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5135–5143).

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., & Kavukcuoglu, K. (2016). Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*.

Kendall, J., Pantone, R., Manickavasagam, K., Bengio, Y., & Scellier, B. (2020). Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*.

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors*. Unpublished doctoral dissertation, Master's Thesis (in Finnish), Univ. Helsinki.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks*, *10*(9), 1659–1671.

Ororbia, A. (2019). Spiking neural predictive coding for continual learning from data streams. *arXiv preprint arXiv:1908.08655*.

Ororbia, A., & Kifer, D. (2022). The neural coding framework for learning generative models. *Nature communications*, *13*(1), 1–14.

Ororbia, A., Mali, A., Giles, C. L., & Kifer, D. (2022). Lifelong neural predictive coding: Learning cumulatively online without forgetting. *Advances in Neural Information Processing Systems*, *35*, 5867–5881.

Ororbia, A. G., Haffner, P., Reitter, D., & Giles, C. L. (2017). Learning to adapt by minimizing discrepancy. *arXiv preprint arXiv:1711.11542*.

Ororbia, A. G., & Mali, A. (2019). Biologically motivated algorithms for propagating local target representations. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 4651–4658).

Ororbia, A. G., Mali, A., Kifer, D., & Giles, C. L. (2018). Deep credit assignment by aligning local representations. *arXiv preprint arXiv:1803.01834*.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., ... Dean, J. (2021). Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Rainer, G., Rao, S. C., & Miller, E. K. (1999). Prospective coding for objects in primate prefrontal cortex. *Journal of Neuroscience*, *19*(13), 5493–5505.

Rao, R. P., & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, *2*(1).

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10684–10695).

Roy, K., Jaiswal, A., & Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature*, *575*(7784), 607–617.

Salvatori, T., Song, Y., Hong, Y., Sha, L., Frieder, S., Xu, Z., ... Lukasiewicz, T. (2021). Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, *34*, 3874–3886.

Shepherd, G. M. (1990). The significance of real neuron architectures for neural network simulations. *Computational neuroscience*, 82–96.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Lanctot, e. a., Marc (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484–489.

Werbos, P. J. (1981). Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization: Proceedings of the 10th ifip conference new york city, usa, august 31–september 4, 1981* (pp. 762–770).