On the role of feedback in image recognition under noise and adversarial attacks: A predictive coding perspective

Andrea Alamia, Milad Mozafari, Bhavin Choksi, Rufin Van-Rullen

On the role of feedback in visual processing: a predictive coding perspective

Andrea Alamia*
CerCo, CNRS, 31052 Toulouse, France
andrea.alamia@cnrs.fr

Milad Mozafari*
CerCo, CNRS, 31052 Toulouse, France
IRIT, CNRS, 31062, Toulouse, France
milad.mozafari@cnrs.fr

Bhavin Choksi
CerCo, CNRS, 31052 Toulouse, France
bhavin.choksi@cnrs.fr

Rufin VanRullen
CerCo, CNRS, 31052 Toulouse, France
ANITI, Université de Toulouse, 31062, France
rufin.vanrullen@cnrs.fr

Abstract

Brain-inspired machine learning is gaining increasing consideration, particularly in computer vision. Several studies investigated the inclusion of top-down feedback connections in convolutional networks; however, it remains unclear how and when these connections are functionally helpful. Here we address this question in the context of object recognition under noisy conditions. We consider deep convolutional networks (CNNs) as models of feed-forward visual processing and implement Predictive Coding (PC) dynamics through feedback connections (predictive feedback) trained for reconstruction or classification of clean images. To directly assess the computational role of predictive feedback in various experimental situations, we optimize and interpret the hyper-parameters controlling the network's recurrent dynamics. That is, we let the optimization process determine whether top-down connections and predictive coding dynamics are functionally beneficial. Across different model depths and architectures (3-layer CNN, ResNet18, and EfficientNetB0) and against various types of noise (CIFAR100-C), we find that the network increasingly relies on top-down predictions as the noise level increases; in deeper networks, this effect is most prominent at lower layers. In addition, the accuracy of the network implementing PC dynamics significantly increases over time-steps, compared to its equivalent forward network. All in all, our results provide novel insights relevant to Neuroscience by confirming the computational role of feedback connections in sensory systems, and to Machine Learning by revealing how these can improve the robustness of current vision models.

1 Introduction

Feed-forward deep convolutional networks (DCNs) reached remarkable accuracy in several visual tasks, including image classification. Interestingly, DCNs share several similarities with biological visual systems. For example, both systems have a hierarchical structure, in which neurons in the higher (lower) levels of the hierarchy have larger (smaller) receptive field sizes and respond to more complex (simpler) stimuli [1]. Further, representational [2] and functional similarities [3] between the feed-forward DCNs and the brain's feed-forward visual pathway have provided novel opportunities to study the brain through the lens of DCNs.

*Equal Contribution

Preprint. Under review.

However, contrary to biological visual systems, DCNs blunder significantly when confronted with noisy images and adversarial attacks, revealing an important deficit in robustness [4, 5, 6]. One main difference with their biological counterpart consists in the lack of recurrent or feedback connections. It has been shown that the brain relies on feedback pathways for robust object recognition under challenging conditions [7, 8, 9, 10, 11]. In recent years, several approaches aimed to introduce feedback connections in deep networks to improve not only biological plausibility but also model robustness, and accuracy [12, 13, 14, 15]. Importantly, feedback connections can be trained either in a supervised fashion to optimize the task objective (e.g., object recognition) or in an unsupervised way to minimize the reconstruction errors (i.e., prediction errors). In the latter case, feedback connections are trained to predict the activity of lower layers, and the network can be described as a hierarchical generative model. More generally, top-down predictions represent prior expectations about lower layers activity, updated based on the incoming sensory evidence over iterations. This interpretation about the role of top-down connections finds its natural place in a prominent framework in Neuroscience, namely Predictive Coding [16, 17].

The Predictive Coding (PC) paradigm in Neuroscience is endorsed by a large body of neuroscientific experimental evidence [18, 19, 20, 21]. It characterizes perception as an inference process in which sensory information is combined with prior expectations to attain the final percept. Accordingly, PC postulates two fundamental terms: predictions and prediction errors (PEs). Considering the visual system as a hierarchical structure, these two signals interact between subsequent brain regions in an iterative process. Ideally, the interplay between feedback predictions and feed-forward PEs converges over iterations into a state in which predictions fully represent the sensory information and PE falls to zero. Although several models implemented and described this dynamic in different conditions [22, 23, 24], the functional role of these two main actors remains largely unexplored.

Here, we address this question by taking a computational perspective and leveraging current state-of-the-art deep neural networks used in visual object recognition. The key insight in our approach consists in letting the network decide for itself (through hyper-parameter optimization) whether top-down connections are functionally beneficial; we then evaluate the outcome across various experimental (noise) conditions. On the one hand, from a Neuroscience point of view, our results supported the hypothesis that feedback plays a crucial role in the cortical processes involved in biological vision. On the other hand, from a machine learning perspective, our simulations demonstrated a more robust class of models based on an established biologically inspired framework.

2 Methods

2.1 Predictive Coding Dynamics

Irrespective of the considered architecture, we implemented the proposed predictive coding dynamics through a stack of modules called *PCoder*s. The activity of each PCoder m_i at time-step t is driven by four terms, as described in the following equation:

$$m_i(t+1) = \mu m_i(t) + \gamma \mathcal{F}_i(m_{i-1}(t+1), \theta_i^{ff}) + \beta \mathcal{B}_{i+1}(m_{i+1}(t), \theta_{i+1}^{fb}) - \alpha \nabla \epsilon_i(t), \quad (1)$$

$$\epsilon_i(t) = \mathit{MSE}(\mathcal{B}_i(m_i(t), \theta_i^{fb}), m_{i-1}(t)), \quad (2)$$

where $\mathcal{F}_i$ computes the feed-forward drive of the ith PCoder with parameters θ_i^{ff}, and $\mathcal{B}_{i+1}$ computes the feedback drive (prediction) with parameters θ_{i+1}^{fb} given m_{i+1}. The gradient $\nabla \epsilon_i(t)$ is calculated with respect to the activity of the higher layer ($m_i(t)$) as suggested in predictive coding theory.

A specific hyper-parameter modulates each term. First, each PCoder's activity is initialized by a feed-forward pass, i.e., without considering memory or top-down connections, in line with experimental observations in biological visual systems [25, 26]. Then, at successive time-steps, the activity is determined by several terms. First, a memory term, regulated by the μ hyper-parameter, that retains information from previous time-steps, essentially acting as a time constant. The γ and β hyper-parameters modulate the feed-forward drive and feedback error terms, which reflect information from the lower and higher layers, respectively. The modulation of the first three terms is normalized, i.e. $\beta + \gamma + \mu = 1$. Lastly, the α hyper-parameter modulates the feed-forward error term, which

aims at reducing the prediction-error, i.e. the mean squared error (MSE) between the prediction by a PCoder and the activity of the lower one (or the "input stimuli" in case of the first PCoder). As an implementation detail, we multiply α by a scaling factor (see Appendix A.2) to remove the effect of batch, layer, and (de)convolution kernel size. As postulated by predictive coding formulation, the feedback and feed-forward error terms regulate each PCoder's activity to reduce prediction-errors over time. Importantly, the dynamic described above is equivalent to the one proposed by Rao and Ballard in 1999 [17], with the only difference being the feed-forward term (for the mathematical proof see [27]).

2.2 Architectures

Shallow model We first implemented a shallow three-layer CNN with two additional dense layers having 120 and 10 neurons, respectively. As shown in figure 1A, the convolutional layers have 12, 18 and 24 channels and a kernel size equal to 5×5. Max-pooling operations with stride equal to 2 were applied from lower to higher layers. In this network, we consider each convolutional layer as a PCoder which predicts the lower one's activity through a bilinear upsampling operation with scale factor equal to 2, followed by a transposed convolutional layer with window size equal to 3×3. The number of channels for the transposed convolution is set in accordance to the prediction target.

Extending to Deep Architectures Given a very deep architecture, it is not computationally efficient to have every layer predicting the preceding one. Instead, we decided to add PC dynamics to blocks of layers (i.e. each PCoder is a sequence of layers). We took advantage of "Predify", a python package introduced in [27], that allows to introduce PC dynamics in pre-trained feed-forward networks. In the present paper, we introduce PResNet18 and PEffNetB0 by adding the proposed PC dynamics to feed-forward ResNet18 and EfficientNetB0 architectures, respectively.

To explore more diversity over input images and network depth, we examined PResNet18 and PEffNetB0 on CIFAR100 and ImageNet, respectively. For PEffNetB0 we used the original EfficientNetB0 architecture with pretrained weights on ImageNet as the feed-forward backbone; However, in order to improve ResNet18 performance on small CIFAR100 images, we lowered the kernel size of the first convolutional layer to 3×3 and omitted its following max-pooling layer to prevent information loss in early layers.

We implemented the block-wise PC dynamics into ResNet18 and EfficientNetB0 by splitting their layers into five and eight PCoders, respectively (see supplementary section A.1). Regardless of the feed-forward architecture, we used a general procedure to define the feed-forward ($\mathcal{F}$) and feedback ($\mathcal{B}$) drive modules. Assume that there are n blocks of layers in the feed-forward network. Let $y = f_i(x)$ denote the computation done by block i where x and y have the size (c_{in}, h_{in}, w_{in}) and $(c_{out}, h_{out}, w_{out})$, respectively. Then, $\mathcal{F}_i$ is f_i and $\mathcal{B}_i$ is a 2D up-scaling operation by the factor of $(h_{in}/h_{out}, w_{in}/w_{out})$ followed by a transposed convolutional layer with c_{out} channels and 3×3 window size.

2.3 Training Parameters

Supervised feed-forward In both shallow and deep models we trained the feed-forward (θ_i^{ff}) and feedback (θ_i^{fb}) parameters separately with different loss functions. First, we trained θ_i^{ff} to optimize the cross-entropy loss (classification) without using the iterative PC dynamics (i.e., in one forward pass). Accordingly, we used a cross-entropy loss with Stochastic Gradient Descent (SGD) optimizer for the shallow model with learning rate 0.01 and momentum 0.9. In the case of deep networks, we trained the modified ResNet18 on CIFAR100 training images for 200 epochs using SGD optimizer with initial learning rate 0.1, momentum 0.9, and weight decay 5e-4. We applied learning rate decay factor 0.2 at epochs 60, 120, and 160. For PeffNetB0, we used the pretrained ImageNet model described in [28].

Unsupervised feedback Next, we optimized θ_i^{fb}s with reconstruction objectives, that is the MSE between the activity of PCoders and their top-down reconstruction on the next time-step. This unsupervised approach is akin to a generative process, in which higher layers predict the activity of lower layers, in line with the predictive coding framework. For the shallow network we used an SGD

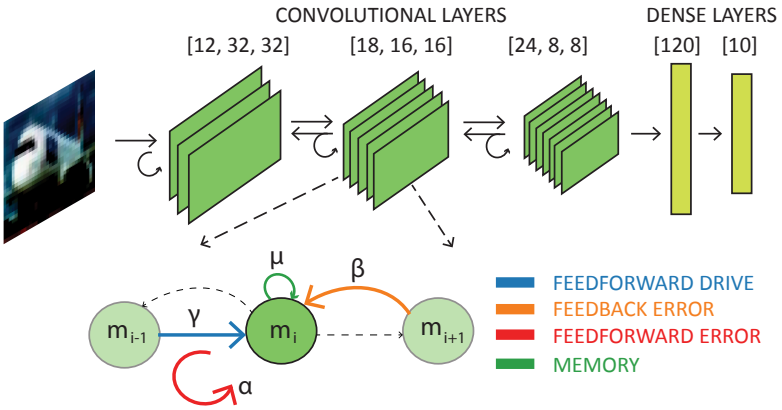

Figure 1: Shallow model architecture and Predictive Coding dynamics. The upper part shows the architecture of the shallow model, composed of three convolutional layers and two fully connected ones. According to Predictive Coding dynamics, the convolutional layers' activity is regulated by four terms, each one modulated by a specific hyper-parameter.

optimizer with learning rate 0.01 and momentum 0.9. While for both of the deep architectures, we employed Adam [29] optimizer with learning rate 0.001 and weight decay 5e-4 for 50 epochs.

Supervised feedback In the shallow model, we also explored the role of the top-down connections when their parameters are trained for classification rather than reconstruction (as in the previous case). In this case both the θ_i^{ff} and θ_i^{fb} are optimized simultaneously for 10 time-steps to minimize the cross-entropy loss. We used an SGD optimizer with learning rate = 0.005 and momentum = 0.9. Since the learning takes place over time-steps, the network optimizes the weights given the PC dynamics described in equation 1. Importantly, during learning we kept the hyper-parameters values to $\gamma = \beta = \mu = 1/3$ and $\alpha = 0.01$.

2.4 Training Hyper-Parameters

After the training of the network's parameters, we froze them (including the statistics of batch normalization layers) and optimized uniquely the hyper-parameters γ, β and α (with μ constrained to be $1 - \beta - \gamma$, see Appendix A.2). Particularly, we repeated the optimization multiple times with different noise types and levels, to investigate the role of each term given different levels of perturbation. We considered a Cross-Entropy loss function averaged across time-steps. In the shallow model we used an Adam optimizer with learning rate equal to 0.001, a weight decay equal to 5e-4 and a batch size of 128 images. For each noise type and level, we repeated the experiment with 10 random initializations of each hyper-parameter drawn from the uniform probability distribution in the interval $[0, 1]$. We used Adam optimizer with the same weight decay for deep models; however, we employed two separate learning rates equal to 0.01 for γ and λ, and 0.0001 for α. We set batch-size to 128 and 16 for PResNet18 and PEffNetB0, respectively. All the scripts and the trained parameters of the main experiments are available on GitHub[2].

2.5 Stimuli

The parameters of both the shallow and the deeper networks were trained on clean images, using CIFAR-10, CIFAR-100 and ImageNet. The hyper-parameters were optimized using different levels and types of noise. Regarding the shallow model, we used additive Gaussian and Salt&Pepper noise, spanning 3 different levels (Gaussian: σ = 0.2, 0.4 and 0.8; Salt&Pepper: pixel percentage = 2%, 4% and 8%). We used CIFAR100-C, a dataset containing five levels of 19 different corruption types [6] to train PResNet18's hyper-parameters. Finally, in order to train hyper-parameters of the deep PEffNetB0, we used the ImageNet validation set and applied five levels of Gaussian (σ= 0.5, 0.75, 1, 1.25, and 1.5) and Salt&Pepper (percentage = 5%, 10%, 15%, 20%, and 30%) noise.

[2] `https://github.com/artipago/Role_of_Feedback_in_Predictive_Coding`

3 Results

3.1 Three-Layer Model

We first tested our hypothesis on a shallow model composed of three convolutional and three dense layers (see panel A of figure 1). The advantage of choosing a smaller network consists in promptly exploring several approaches before replicating in deeper state-of-the-art networks. Specifically, we investigated the role of each term in equation (1): 1) when training feedback weights for reconstruction or classification (unsupervised vs supervised), 2) via some ablation simulations, and 3) regarding the robustness to adversarial attacks.

3.1.1 Feedback weights: reconstruction vs. classification

We first assessed the role of the feedback and each term in equation (1) when the top-down parameters were optimized for reconstruction. After having trained the forward weights for classification (Supplementary figure 5A), we trained the feedback weights optimizing the reconstruction loss of each PCoder (figure 5C). This approach is in line with the PC interpretation, in which top-down connections generate predictions to explain lower layers' activity (i.e., minimize prediction errors, or the reconstruction loss). In this case backward weights are trained in an unsupervised fashion. Once both forward and backward connections were optimized (for classification and reconstruction, respectively), we froze all parameters and trained only the hyper-parameters (γ, β and α in equation 1). As shown in figure 2A, with both Gaussian and Salt&Pepper noise the hyper-parameter modulating the top-down feedback (i.e., β in equation 1) increases as a function of the noise level, supporting the hypothesis that top-down connections are crucial for visual processing in noisy conditions. Remarkably, also α, which modulates the amount of bottom-up prediction-error, increases with the noise level for both types of noise. Similar results were obtained when training the top-down parameters for classification rather than reconstruction (i.e., supervised approach). As in the unsupervised case, when freezing the parameters and optimizing exclusively the hyper-parameters for different noise levels, we observed an increase of both bottom-up (α) and top-down (β) errors as a function of the noise level. Yet, figure 2C shows that top-down parameters trained for reconstruction proved more robust to noisy images than those trained for classification. Next, we compared the networks' performance with equivalent forward networks. First, we trained (on clean images) four types of forward networks: either having the same forward architecture as the shallow network (labeled "same" in figure 2B, and resulting in a slightly smaller number of parameters), or having a larger number of parameters by increasing either the kernel size, or the number of features, or the layers (labeled "kernel", "feat" and "deep", respectively). As summarized in figure 2B, both networks implementing predictive coding dynamics (in cyan and green in the figure) perform systematically better than all the forward networks, irrespective of the noise type and level. This result demonstrates that feedback connections, and specifically predictive coding dynamics, improve network robustness to noise.

3.1.2 Ablation studies

We then investigated how selectively removing the top-down or the bottom-up error term influences the results. Importantly, we focused specifically on the unsupervised network, whose top-down parameters are trained for reconstruction, and that better represents the PC dynamics. As shown in figure 2A, when removing the top-down error term, the forward error hyper-parameter increases with the noise levels and doubles its value as compared to the full model (labeled "unsupervised" in the figure). On the other hand, when removing the forward error term, we observed an increase of the feedback term with the noise levels, as in the full model. Concerning the networks performance, figure 2C reveals that removing the top-down feedback degrades the accuracy with higher noise levels (especially with Gaussian noise), confirming the conclusion that top-down feedback plays a crucial role in the processing of degraded images.

3.1.3 Adversarial attacks

To further confirm the hypothesis that top-down feedback is important for robustness, we froze the networks (feedforward, full predictive coding, or ablated networks) with manual configurations of the hyper-parameters and then tested their robustness against targeted L_∞ Random Projected Gradient Descent (RPGD) [30] and Basic Iterative Method (BIM) [31] attacks, after unrolling

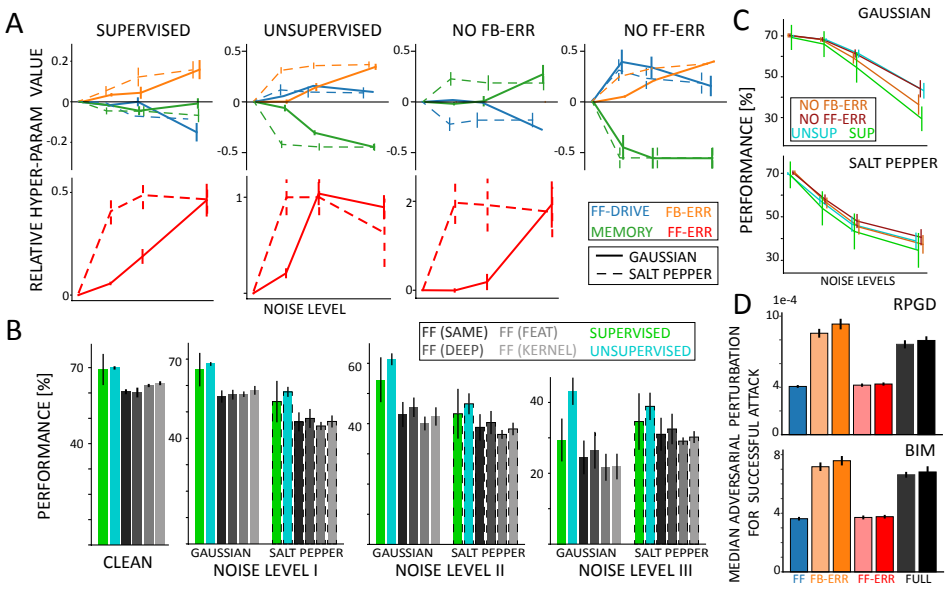

Figure 2: Shallow model results. A) The plots show the hyper-parameters (HPs) value relative to the clean images as a function of the noise levels. Each column shows the relative HPs trained in different conditions: supervised, unsupervised, without feedback error, or forward error. The first row shows the feedback error term, the memory, and the forward drive term, the second row shows the forward error term on a separate scale, for Gaussian (solid line) and Salt&Pepper (dashed lines) noise. In all conditions, the feedback-error and forward-error terms increase with the noise levels. B) The models implementing PC dynamics (in green and cyan) perform better than equivalent feed-forward networks, especially when trained in an unsupervised fashion (cyan). C) Performance of the PC models, as a function of the noise levels, measured at the last time-step. Contrasting supervised (SUP) and unsupervised (UNSUP) models reveals the effects of feedback training objective, whereas comparing the ablation models with UNSUP shows the effect of each error term on accuracy. D) The graph shows the median perturbation to obtain a successful attack using different HPs. Orange and red bars have higher feedback and forward error terms (paler colors correspond to smaller error terms), and blue and black bars represent the feed-forward and the full model, respectively. Our simulations reveal that PC models with higher feedback values (orange, black bars) are more robust to adversarial perturbations.

them for 10 time-steps to keep their depths constant. We use Foolbox API 2.4.0 [32] and measure the median perturbation required to successfully fool the networks. As shown in figure 2D, we observe that networks with higher top-down feedback (two orange bars in the figure have $\alpha = 0$ and $\gamma = \beta = \mu = 0.33$; and $\beta = 0.5, \gamma = 0.3, \mu = 0.2$, respectively) reveal better robustness to the attacks as compared to the equivalent forward network (in blue in the figure, with $\gamma = 1$ and all other hyper-parameters set to zero). Interestingly, a forward network leveraging only the feed-forward error shows a similar (lack of) robustness to the attack as the forward network (in red in the figure, both networks having $\gamma = 1$, and $\alpha = 1$ and $\alpha = 2$, respectively; all other hyper-parameters set to zero). Additionally, adding the feed-forward error to the model with top-down connections, slightly reduces its robustness (black bars in the picture, both networks with $\alpha = 1$ and $\gamma = 0.3$, while $\beta = \mu = 0.33$ and $\beta = 0.5$ $\mu = 0.2$, respectively). These results confirm that top-down connections are useful for adversarial robustness (as shown on a different dataset with a different PC implementation by Huang and colleagues [12]), but also suggest that feedforward error correction does not help adversarial robustness. This is likely because the feedforward prediction errors emphasize the input perturbation, which the generative feedback was not trained to account for.

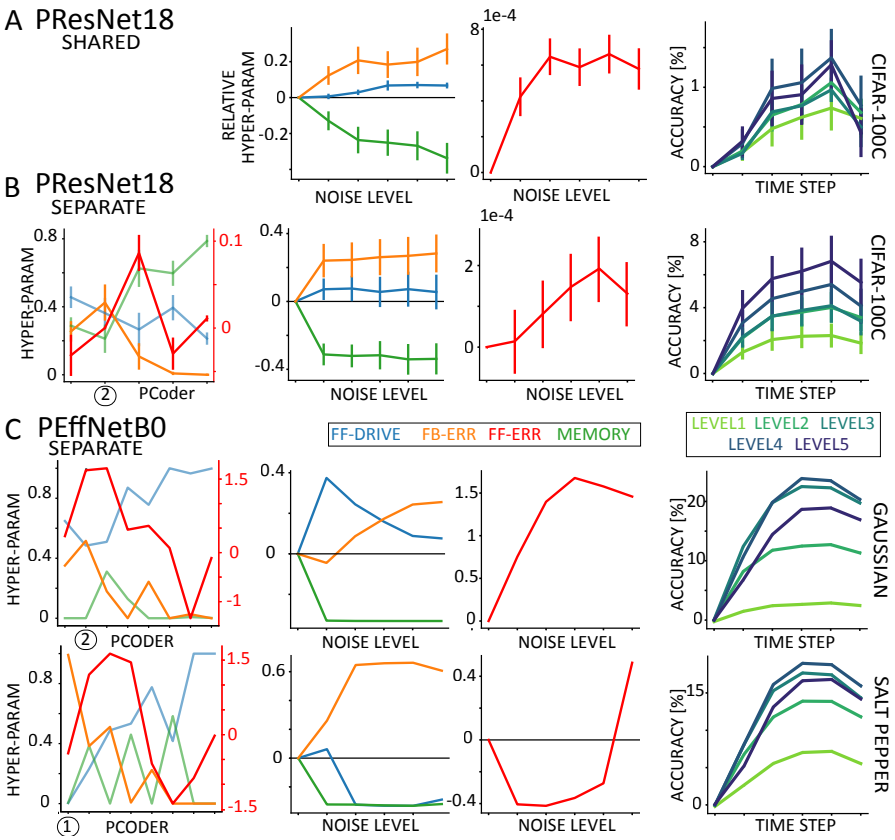

Figure 3: Values of hyper-parameters and accuracy of the deep predictive coding networks. (A) PResNet18 with shared hyper-parameters that are trained on CIFAR100-C images. (B) PResNet18 and (C) PEffNetB0 with separate hyper-parameters that are trained respectively on CIFAR100-C and ImageNet under Gaussian and Salt&Pepper noise. Plots in the first column show the hyper-parameters as a function of PCoders under medium noise level. The circles indicate PCoders with maximum feedback error. In middle columns, relative values of hyper-parameters are plotted across noise levels. In case of separate hyper-parameters, the PCoder with maximum feedback error is shown. Accuracy change for each noise level is depicted in the last column. Error bars show standard error of the mean (SEM) over 19 CIFAR100-C noise types. In all cases, the networks achieve accuracy gain by utilizing more feedback and forward error as the noise severity increases. See supplementary Figures 7-12 for the absolute values of hyper-parameters and changes in recognition accuracy per noise type and level.

3.2 Deep Models

3.2.1 Shared hyper-parameters

Similar to the three-layer network, we examined PResNet18 with a single set of α, β, and γ, that is shared between all the PCoders. In this experiment, we followed the unsupervised training approach explained for the three-layer network using the CIFAR100 dataset. After having optimized the top-down connections for reconstruction, we froze the weights and trained the hyper-parameters to minimize the average cross-entropy loss over five time-steps. We performed this optimization independently on each noise type and noise level of the CIFAR100-C dataset.

Figure 3A shows the average hyper-parameter values across all 19 noise types relative to those learned using "clean" images. Confirming the results of the shallow model, we observed that the roles of feedback and feed-forward error become more crucial as the noise level increases. Importantly, the average accuracy change across time-steps reveals a very robust (but marginal) improvement with respect to the feed-forward ResNet18 for all levels of noise. Remarkably, the importance of feedback connections shines more as the noise severity increases.

3.2.2 Separate hyper-parameters

Encouraged by the results in the "shared" approach described above, we decided to provide each PCoder with a separate set of hyper-parameters. Our reasoning was that different stages of the hierarchical visual processing would benefit differently from the combination of top-down and bottom-up information, thus granting to the network more flexibility in accounting for different representations across different layers.

As in the previous experiment, we trained PResNet18's hyper-parameters on CIFAR100-C images. Moreover, in order to validate our previous results on a more complex dataset, we trained PEffNetB0's hyper-parameters on the ImageNet2012 validation set for five levels of Gaussian and Salt&Pepper noises.

Introducing a separate set of hyper-parameters in each PCoder resulted in a very significant boost in recognition accuracy of both networks, under all conditions. As illustrated in the last column of Figure 3B, PResNet18 consistently improved the recognition accuracy across time-steps on all noise types and levels, revealing an average improvement around 6% in the most noisy condition. Remarkably, we could replicate these results using the deeper network PEffNetB0 with eight PCoders. As shown in Figure 3C, PC dynamics with different hyper-parameters per PCoder yielded an impressive increase in accuracy above 20% and above 15% in the worst condition of Gaussian and Salt&Pepper noise, respectively.

We then investigated the trend of hyper-parameters across PCoders. This analysis shed some light on the role of the hyper-parameters as a function of their hierarchical stage in the network. Remarkably, we obtained very consistent results on both networks, and across different noise types. The first column in panels B and C of Figure 3 shows the values of hyper-parameters as a function of PCoders for the medium noise level (level 3, results don't change across noise levels, see supplementary figures 10-12). Regardless of the considered model, we found that the PCoder with the largest amount of feedback error hyper-parameter (indicated by a circle in the figure), is consistently situated at the lower layers of the network, whereas the feedback tends to zero at higher layers. This suggests that the beneficial effects of top-down connections are best achieved at lower layers of the visual hierarchy, where high-level expectations shape low level features to maximize the final classification.

In addition, the second and third columns of Figure 3B, C confirmed our previous results, revealing how the feedback-error term increases as a function of the noise levels in the PCoder with its highest values (i.e., the second for PResNet18, and either the first or the second in PEffNetB0, depending on the noise type). This result confirms once again the hypothesis that robust object recognition requires more top-down influence (i.e., feedback and feed-forward error terms) as the level of noise increases.

4 Discussion

4.1 Summary of the Results

Starting from an established framework in Neuroscience, namely Predictive Coding (PC), we investigated the role of top-down feedback connections in models of vision. The significance of our work spans across Neuroscience and machine learning, contributing substantially to both fields. First, our results demonstrated how predictive coding dynamics increase the network's robustness to various types of noisy stimuli compared to equivalent feed-forward networks. Additionally, systematic optimization of hyper-parameters revealed how the feedback contribution increases with the noise severity, especially in the early stages of the network, providing important information about the role of top-down processes in visual processing. Compared with prior studies, one original aspect of our approach is our empirical procedure, in which we let the optimization process converge to the optimal solution in each noise level.

4.2 Previous Work

Previous studies explored the supervised approach to train feedback connections for classification rather than reconstruction objectives. Feedback Networks [33] introduced top-down and temporal skip connections in a recurrent convolutional module, demonstrating an increase in performance followed by improvements in early features representation, taxonomic predictions, and curriculum learning. Similarly, Nayebi and colleagues [14] proposed a ConvRNN architecture, incorporating gating and

skip connections, which significantly improved object recognition performance. Considering models advocating more explicitly for biological plausibility, Linsley and colleagues [34] suggested another recurrent vision model, equipped with horizontal and gated recurrent units (hGRU). Its performance improves specifically in recognition tasks involving long-range spatial dependencies. Supported by experimental studies [9], Kubilius and colleagues also proposed a brain-inspired architecture named CorNet, which includes feedback and skip connections. Interestingly, it reveals high neural similarity to cortical visual areas such as V4 and IT [13].

In the PC domain, Chalasani and Principe [35] proposed a hierarchical, generative model based on PC dynamics, including context-sensitive priors on the latent representations. Their architecture demonstrated how top-down connections from higher layers are instrumental in solving lower layers ambiguities, providing some noise robustness. The model proposed in [36] is the closest one to ours. Despite following PC dynamics and the principal similarities, their model presents some critical limitations. More specifically, all weights are trained for object recognition only at the last time step, resulting in a biologically implausible behavior, in which near-chance performance is observed until the final iteration. A more in-depth comparison between this work and our proposed method is presented in [27]. Finally, Huang and colleagues [12] implemented unsupervised feedback connections by optimizing for "self consistency" between the input image features, latent variables and label distribution. Despite a different dynamics, PC principles inspired their implementation, which also provided some robustness against gradient-based adversarial attacks on Fashion-MNIST and CIFAR10.

4.3 Insights for and from Neuroscience

It is possible to characterize the role of top-down feedback either as an unsupervised, generative process which predicts lower layers' activities, or as a supervised, discriminative process to optimize classification. Besides being more biologically plausible, our simulations with the shallow model revealed that the unsupervised approach is more robust to noise than the supervised one, as shown in figure 2B. However, when trained with supervision, feedback connections do not converge to the unsupervised solution, as shown in figure 5C which compares the reconstruction errors in shallow models optimized for classification (supervised) or reconstruction (unsupervised).

Interestingly, when we independently optimized each PCoder in deeper networks (roughly equivalent to different brain regions across the hierarchy of visual processes), we observed consistently higher modulation of top-down activity in lower regions, and relatively less top-down feedback in higher areas. Choksi and Mozafari et al. [27] further demonstrates that the proposed biologically-inspired feedback dynamics iteratively project the noisy inputs towards the learned data manifold, similar to previous studies using different approaches [37, 38, 39, 40]. Future research may test this prediction directly in biological brains by recording the top-down cortical activity at different stages of the visual hierarchy, and validate the hypothesis that early brain regions benefit the most from top-down feedback during visual perception in noisy conditions.

Our results demonstrated how top-down and bottom-up processes influence perception in different challenging conditions. However, how does the brain modulate each term's contribution (i.e., each hyper-parameter) during natural vision? Attention mechanisms may be responsible for the regulation of top-down processes by increasing feedback response during noisy conditions [41, 42]. Accordingly, it could be possible to envision a model inspired by current transformer architectures where a biologically plausible attention system modulates hyper-parameters based on input features or top-down expectations [43]. Expectation is another important process that modulates top-down feedback in the human brain [44, 45]. In our model, the forward pass initializes the activity in each layer based on the first processing of the input (i.e., without the recurrent PC dynamic). However, it is possible to initialize the network's activity based on top-down beliefs, according to PC dynamics: the last layer of the hierarchy encodes the predictions of the expected input (i.e., a given class in a classification dataset), and propagates such predictions to initialize the activity of lower layers, similarly to the brain processes involved in sensory expectations [46, 47]. Future work could explore how such expectations may influence the network behavior and accuracy.

Broader Impact

Despite its outstanding achievements, artificial intelligence (AI) revealed significant reliability limitations when tested in challenging conditions. Addressing these concerns is becoming a crucial goal for the scientific community, as AI is gaining an important place in our daily lives. In this work, we leverage an established framework in Neuroscience –namely Predictive Coding- to address this problem and investigate the role of feedback in robust visual processing. Our results suggested that inspiration from the human brain can be beneficial for artificial vision, and our model provides a remarkable tool to study the visual system in biological brains [48]. On the one hand brain-inspired approaches can boost artificial sensory processes in ecological (noisy) situations. On the other hand, we are aware of the possible nefarious use of human-like artificial systems, and we encourage researchers and policymakers to consider these issues and their societal implications.

Acknowledgments and Disclosure of Funding

RV is supported by an ANITI (Artificial and Natural Intelligence Toulouse Institute) Research Chair (grant ANR-19-PI3A-0004), and two ANR grants AI-REPS (ANR-18-CE37-0007-01) and OSCI-DEEP (ANR-19-NEUC-0004).

References

[1] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[2] Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915, 2014.

[3] Pouya Bashivan, Kohitij Kar, and James J DiCarlo. Neural population control via deep image synthesis. *Science*, 364(6439), 2019.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[5] A Nguyen, J Yosinski, and J Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. arxiv, cs. *arXiv preprint arXiv:1412.1897*, 2014.

[6] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[7] Dean Wyatte, David J Jilk, and Randall C O'Reilly. Early recurrent feedback facilitates visual object recognition under challenging conditions. *Frontiers in psychology*, 5:674, 2014.

[8] Tim C Kietzmann, Courtney J Spoerer, Lynn KA Sörensen, Radoslaw M Cichy, Olaf Hauk, and Nikolaus Kriegeskorte. Recurrence is required to capture the representational dynamics of the human visual system. *Proceedings of the National Academy of Sciences*, 116(43):21854–21863, 2019.

[9] Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B Issa, and James J DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature neuroscience*, 22(6):974–983, 2019.

[10] Karim Rajaei, Yalda Mohsenzadeh, Reza Ebrahimpour, and Seyed-Mahdi Khaligh-Razavi. Beyond core object recognition: Recurrent processes account for object recognition under occlusion. *PLoS computational biology*, 15(5):e1007001, 2019.

[11] Kohitij Kar and James J DiCarlo. Fast recurrent processing via ventrolateral prefrontal cortex is needed by the primate ventral stream for robust core visual object recognition. *Neuron*, 109(1):164–176, 2021.

[12] Y Huang, J Gornet, S Dai, Z Yu, T Nguyen, DY Tsao, and A Anandkumar. Neural networks with recurrent generative feedback. arxiv, cs. *arXiv preprint arXiv:2007.09200*, 2020.

[13] Jonas Kubilius, Martin Schrimpf, Aran Nayebi, Daniel Bear, Daniel LK Yamins, and James J DiCarlo. Cornet: Modeling the neural mechanisms of core object recognition. *BioRxiv*, page 408385, 2018.

[14] Aran Nayebi, Daniel Bear, Jonas Kubilius, Kohitij Kar, Surya Ganguli, David Sussillo, James J DiCarlo, and Daniel LK Yamins. Task-driven convolutional recurrent models of the visual system. *arXiv preprint arXiv:1807.00053*, 2018.

[15] Siming Yan, Xuyang Fang, Bowen Xiao, Harold Rockwell, Yimeng Zhang, and Tai Sing Lee. Recurrent feedback improves feedforward representations in deep neural networks. *arXiv preprint arXiv:1912.10489*, 2019.

[16] Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

[17] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[18] James M Kilner, Karl J Friston, and Chris D Frith. Predictive coding: an account of the mirror neuron system. *Cognitive processing*, 8(3):159–166, 2007.

[19] Torsten Baldeweg. Repetition effects to sounds: evidence for predictive coding in the auditory system. *Trends in cognitive sciences*, 2006.

[20] Marta I Garrido, James M Kilner, Klaas E Stephan, and Karl J Friston. The mismatch negativity: a review of underlying mechanisms. *Clinical neurophysiology*, 120(3):453–463, 2009.

[21] Jakob Hohwy, Andreas Roepstorff, and Karl Friston. Predictive coding explains binocular rivalry: An epistemological review. *Cognition*, 108(3):687–701, 2008.

[22] Karl Friston and Stefan Kiebel. Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1211–1221, 2009.

[23] Andrea Alamia and Rufin VanRullen. Alpha oscillations and traveling waves: Signatures of predictive coding? *PLoS Biology*, 17(10):e3000487, 2019.

[24] Michael W Spratling. Predictive coding as a model of response properties in cortical area v1. *Journal of neuroscience*, 30(9):3531–3543, 2010.

[25] Rufin VanRullen and Simon J Thorpe. Is it a bird? is it a plane? ultra-rapid visual categorisation of natural and artifactual objects. *Perception*, 30(6):655–668, 2001.

[26] Rufin VanRullen and Simon J Thorpe. The time course of visual processing: from early perception to decision-making. *Journal of cognitive neuroscience*, 13(4):454–461, 2001.

[27] Bhavin Choksi, Milad Mozafari, Callum Biggs O'May, Benjamin Ador, Andrea Alamia, and Rufin VanRullen. Predify: Augmenting deep neural networks with brain-inspired predictive coding dynamics. 2021.

[28] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[31] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[32] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

[33] Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1308–1317, 2017.

[34] Drew Linsley, Junkyung Kim, Vijay Veerabadran, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated-recurrent units. *arXiv preprint arXiv:1805.08315*, 2018.

[35] Rakesh Chalasani and Jose C Principe. Deep predictive coding networks. *arXiv preprint arXiv:1301.3541*, 2013.

[36] Haiguang Wen, Kuan Han, Junxing Shi, Yizhen Zhang, Eugenio Culurciello, and Zhongming Liu. Deep predictive coding network for object recognition. In *International Conference on Machine Learning*, pages 5266–5275. PMLR, 2018.

[37] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147, 2017.

[38] Shiwei Shen, Guoqing Jin, Ke Gao, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan. *arXiv preprint arXiv:1707.05474*, 2017.

[39] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[40] Ajil Jalal, Andrew Ilyas, Constantinos Daskalakis, and Alexandros G Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

[41] Farhan Baluch and Laurent Itti. Mechanisms of top-down attention. *Trends in neurosciences*, 34(4):210–224, 2011.

[42] Harriet Feldman and Karl Friston. Attention, uncertainty, and free-energy. *Frontiers in human neuroscience*, 4:215, 2010.

[43] Rufin VanRullen and Andrea Alamia. Gattanet: Global attention agreement for convolutional neural networks. *arXiv preprint arXiv:2104.05575*, 2021.

[44] Floris P De Lange, Micha Heilbron, and Peter Kok. How do expectations shape perception? *Trends in cognitive sciences*, 22(9):764–779, 2018.

[45] Christopher Summerfield and Floris P De Lange. Expectation in perceptual decision making: neural and computational mechanisms. *Nature Reviews Neuroscience*, 15(11):745–756, 2014.

[46] Christopher Summerfield and Tobias Egner. Expectation (and attention) in visual cognition. *Trends in cognitive sciences*, 13(9):403–409, 2009.

[47] Peter Kok and Floris P de Lange. Predictive coding in sensory cortex. In *An introduction to model-based cognitive neuroscience*, pages 221–244. Springer, 2015.

[48] Zhaoyang Pang, Callum Biggs O'May, Bhavin Choksi, and Rufin VanRullen. Predictive coding feedback results in perceived illusory contours in a recurrent neural network. *arXiv preprint arXiv:2102.01955*, 2021.

A Appendix

A.1 Deep Network Architectures

In this part, we explain how we split each of the ResNet18 and EfficientNetB0 into blocks of layers and converted them into PCoders.

We used a modified ResNet18 architecture that works better with 32x32 images from the CIFAR100 dataset. ResNet18 is a sequence of residual blocks, each of which consists of a sequence of convolution, batch normalization, and ReLU layers. Due to the residual connections around each block, we chose to never split them into multiple PCoders. However, a single PCoder may contain more than one residual blocks in its feedforward module ($\mathcal{F}$). More precisely, we split the modified ResNet18 into 5 PCoders. PCoder1 contains the first Convolution and Batch Normalization layers. PCoder2 to PCoder5 contain two consecutive residual blocks each.

In the case of PEffNetB0, we used the PyTorch implementation of EfficientNetB0 provided in `https://github.com/rwightman/pytorch-image-models`. In this implementation, EfficientNetB0 is split into eight blocks of layers (considering the first convolution and batch normalization layers as a separate block). Except for the classification block, we converted each block into a PCoder.

Please see Table 1 for more details on deep predictive coding architectures and their PCoders.

Table 1: Architectures of PResNet18 and PEffNetB0. Conv (channel, size, stride), Deconv (channel, size, stride), Upsample (scale_factor), BN is BatchNorm, $[\,]_+$ is ReLU, and $[\,]_*$ is SiLU. EfficientBlock corresponds to each block in the PyTorch implementation of EfficientNetB0. See Table 2 for the structure of ResNet BasicBlocks

	PResNet18 Input Size: 3x32x32		PEffNetB0 Input Size: 3x224x224	
	$\mathcal{F}_i$	$\mathcal{B}_i$	$\mathcal{F}_i$	$\mathcal{B}_i$
PCoder1	[BN (Conv (64, 3, 1))]$_+$	Deconv (3, 3, 1)	[BN (Conv (32, 3, 2))]$_*$	Upsample (2) Deconv (3, 3, 1)
PCoder2	[BasicBlock (64, 3, 1)]$_+$ [BasicBlock (64, 3, 1)]$_+$	Deconv (64, 3, 1)	EfficientBlock0	Deconv (32, 3, 1)
PCoder3	[BasicBlock (128, 3, 2)]$_+$ [BasicBlock (128, 3, 1)]$_+$	Upsample (2) Deconv (64, 3, 1)	EfficientBlock1	Upsample (2) Deconv (16, 3, 1)
PCoder4	[BasicBlock (256, 3, 2)]$_+$ [BasicBlock (256, 3, 1)]$_+$	Upsample (2) Deconv (128, 3, 1)	EfficientBlock2	Upsample (2) Deconv (24, 3, 1)
PCoder5	[BasicBlock (512, 3, 2)]$_+$ [BasicBlock (512, 3, 1)]$_+$	Upsample (2) Deconv (256, 3, 1)	EfficientBlock3	Upsample (2) Deconv (40, 3, 1)
PCoder6	-	-	EfficientBlock4	Deconv (80, 3, 1)
PCoder7	-	-	EfficientBlock5	Upsample (2) Deconv (112, 3, 1)
PCoder8	-	-	EfficientBlock6	Deconv (192, 3, 1)

Table 2: Architecture of BasicBlock(i, j, k). Each BasicBlock is a residual block where the input is added to the output of the block. When $k \neq 1$, the input passes through a Conv(i,1,2) and a BatchNorm before being added to the output.

BasicBlock (i, j, k)
[BN (Conv (i, j, k))]$_+$ BN (Conv (i, j, 1))

A.2 Implementation Details

Training Hyper-Parameters Since μ, γ, and λ should satisfy the constraint $\mu + \gamma + \lambda = 1$, independently optimizing them with backpropagation leads to invalid values. To solve this issue, we made use of auxiliary parameters. Precisely, let μ_{aux}, γ_{aux}, and λ_{aux} denote three auxiliary parameters. Then, we compute μ, γ, and λ as follows:

$$\mu = \frac{\sigma(\mu_{aux})}{\sigma(\mu_{aux}) + \sigma(\gamma_{aux}) + \sigma(\beta_{aux})}, \tag{3}$$

$$\gamma = \frac{\sigma(\gamma_{aux})}{\sigma(\mu_{aux}) + \sigma(\gamma_{aux}) + \sigma(\beta_{aux})}, \tag{4}$$

$$\beta = \frac{\sigma(\beta_{aux})}{\sigma(\mu_{aux}) + \sigma(\gamma_{aux}) + \sigma(\beta_{aux})}, \tag{5}$$

where

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{6}$$

While the auxiliary parameters can take on any real value, the corresponding hyper-parameters are thus constrained between 0 and 1, summing to 1.

Gradient Scaling In our dynamics, the error (ϵ_i) is defined as a scalar quantity whose gradient is taken with respect to the activation of the higher layer (m_i). That is,

$$\epsilon_i = \frac{1}{K} \sum_{k}^{K} (m_{i-1}^k - p_{i-1}^k)^2 \tag{7}$$

where p_{i-1} ($= \mathcal{B}(m_i, \theta_{i+1}^{fb})$) represents the prediction made for m_{i-1} and K represents the number of elements in m_{i-1} (= channels*width*height). Thus, the error-correction term at position j itself becomes,

$$\frac{\partial \epsilon_i}{\partial m_i^j} = \frac{1}{K} \sum_{k}^{K} \frac{\partial (m_{i-1}^k - p_{i-1}^k)^2}{\partial m_i^j} \tag{8}$$

Equation 8 highlights how the dimensionality of the prediction (equivalently the error term) affects the gradients, scaling them with a factor K that can differ across layers by orders of magnitude. This effect is worsened for CNNs where the gradients outside of the receptive field (of size C) of element m_i^j will be zero,

$$\sum_{k}^{K} \frac{\partial (m_{i-1}^k - p_{i-1}^k)^2}{\partial m_i^j} = \sum_{k}^{C} \frac{\partial (m_{i-1}^k - p_{i-1}^k)^2}{\partial m_i^j} \tag{9}$$

To counteract this, we apply a layer-specific scaling factor to the error gradients. Assuming that the partial derivative of the error for each pair of connected neurons i, j is i.i.d normally distributed around 0 :

$$\frac{\partial (m_{i-1}^k - p_{i-1}^k)^2}{\partial m_i^j} \sim \mathcal{N}(0, \sigma^2) \tag{10}$$

It can be shown that,

$$\frac{\partial \epsilon_i}{\partial m_i^j} = \frac{1}{K} \sum_{k}^{C} \frac{\partial (m_{i-1}^k - p_{i-1}^k)^2}{\partial m_i^j} \sim \mathcal{N}(0, \frac{C\sigma^2}{K^2}) \tag{11}$$

Equation 11 provides a way to, at least partly, counteract the effect of the dimensionality for our gradient. We multiply the gradient by a factor of $\sqrt{K^2/C}$ to scale them and apply a more meaningful step size for correcting the errors.

Execution Time We tested the shallow models and the deeper networks on different machines. All the simulations of the shallow models, including the 10 different initializations, took approximately 4 days using 1 GPU Nvidia GTX 1080Ti with 11Gb. The training of PResNet18 on CIFAR-C took approximatively 2 weeks, whereas PEffNetB0 was trained in 6 days, using a machine equipped with 1 GPU Nvidia TitanV with 12Gb.

A.3 Supplementary Figures

In this section, we present the supplementary figures which complement the main text. The first three figures (fig. 4-6) show the full results of the shallow model: the value of its hyper-parameters (figure 4), the results concerning the training of its parameters (figure 5), and the accuracy over time steps (figure 6). Figures 7 and 8 show the relative accuracy for PResNet18 on all noise levels in CFIAR100-C, using shared or separate hyperparameters, respectively. The last four figures (fig 9-12) report all the hyper-parameters values for PResNet18 and PEffNetB0, considering different noise levels and -in the case of separate hyper-parameters- each PCoder.

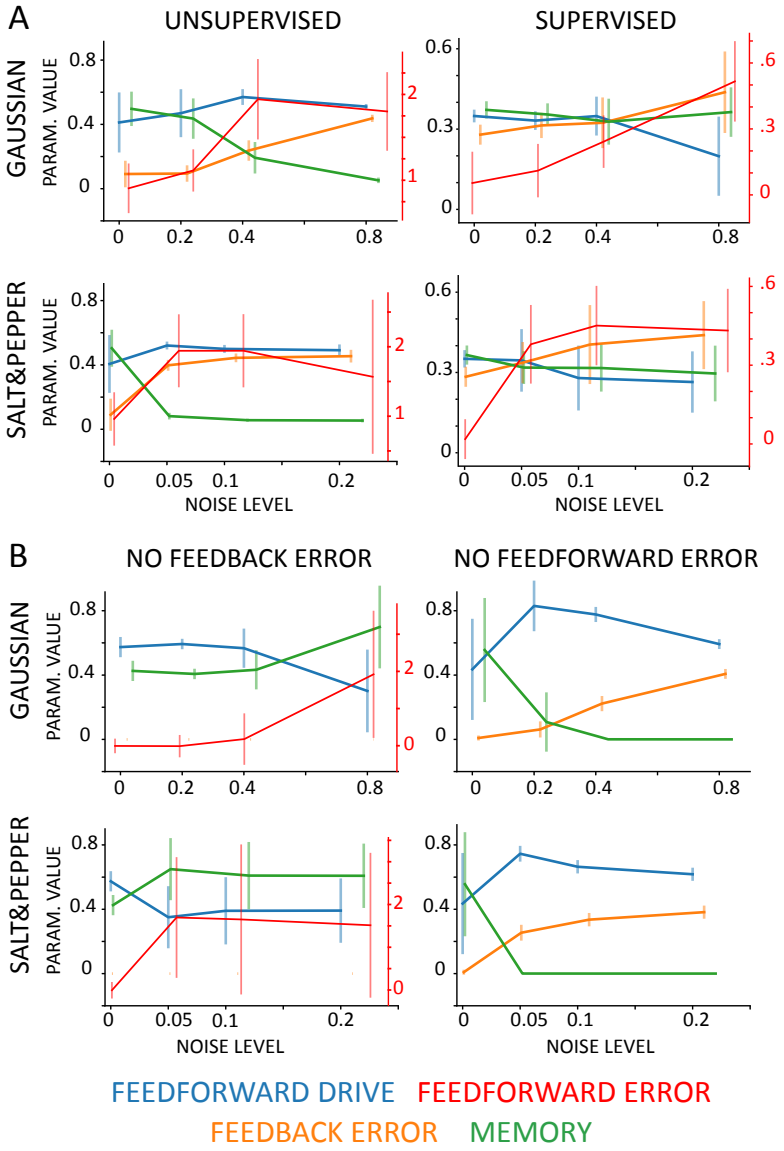

Figure 4: Hyper-parameters for the shallow model. A) Each subplot shows the hyper-parameter values for the unsupervised and supervised networks (left and right column, respectively) for gaussian and salt&pepper noise (first and second row, respectively). The color code is consistent with the main figure and indicated at the bottom of the figure. The feedforward-error refers to its own y-axis shown to the right of each panel. B) Same as in A but for the ablation models, in which either the feedback-error (left column) or the feedforward-error (right column) were removed.

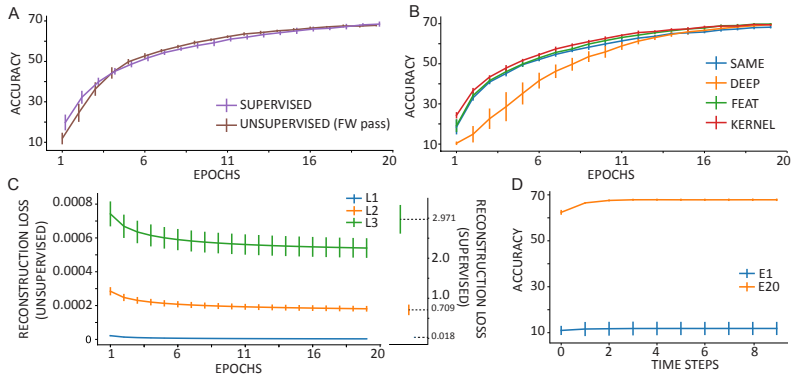

Figure 5: Shallow models training. A) Networks' accuracy during the training. In the supervised case the hyper-parameters were kept fixed, whereas in the unsupervised case we report the accuracy of the forward pass only. B) Training accuracy of the forward networks. C) Reconstruction loss for each Pcoder in the shallow model. For comparison, we report to the right the reconstruction loss in the supervised network (note that in this case the network was not trained for reconstruction). D) Accuracy of the supervised network as a function of the time-steps in the first and last block of training.

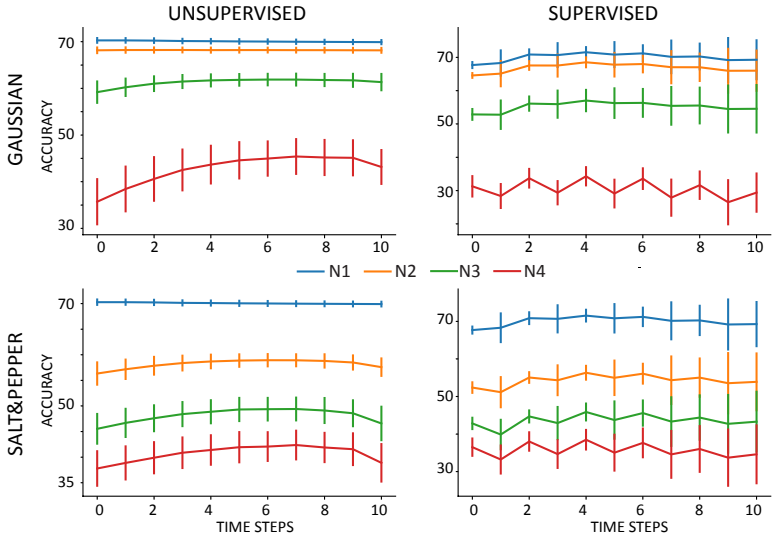

Figure 6: Accuracy over time steps of both shallow models trained for reconstruction (left column) or classification (right column). The colors represent different noise levels for Gaussian (first row) and Salt&pepper noise (second row).

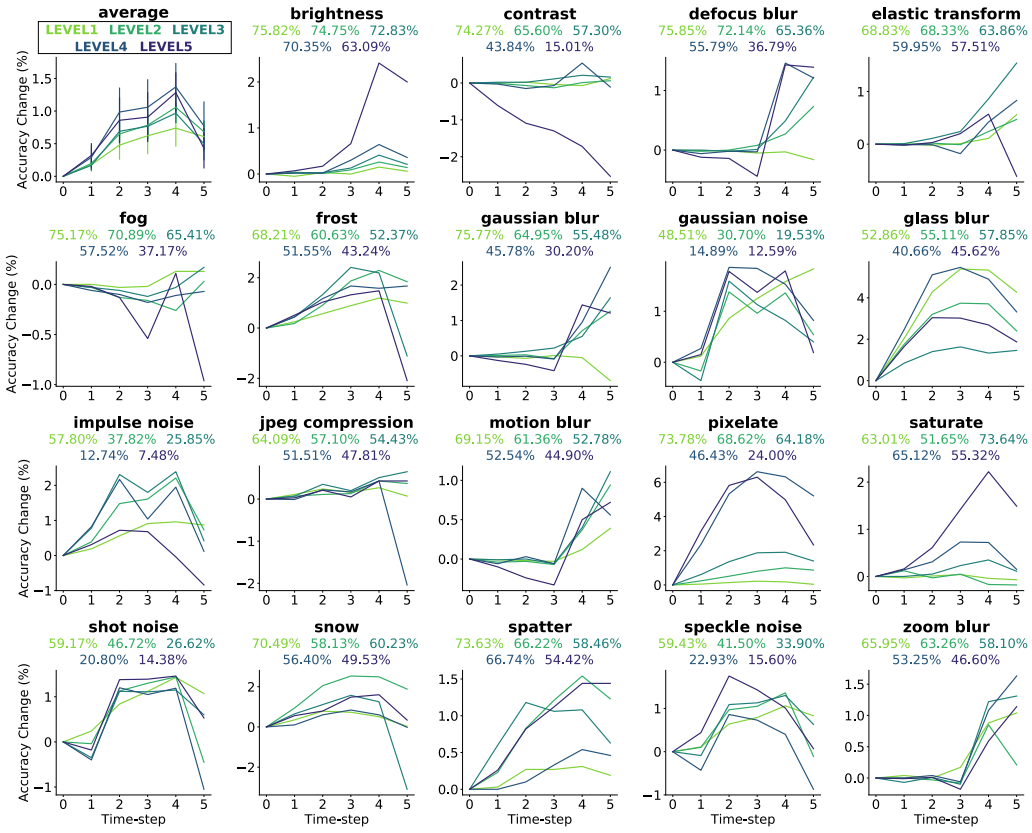

Figure 7: Recognition accuracy of PResNet18 with shared hyper-parameters on each of the CIFAR100-C noise types and levels. Each plot shows the change in accuracy with respect to the feedforward baseline (i.e. ResNet18) for each noise type. Each color indicates a noise level. Numbers below the noise names denote the absolute recognition accuracy at time-step 0 (feedforward baseline).

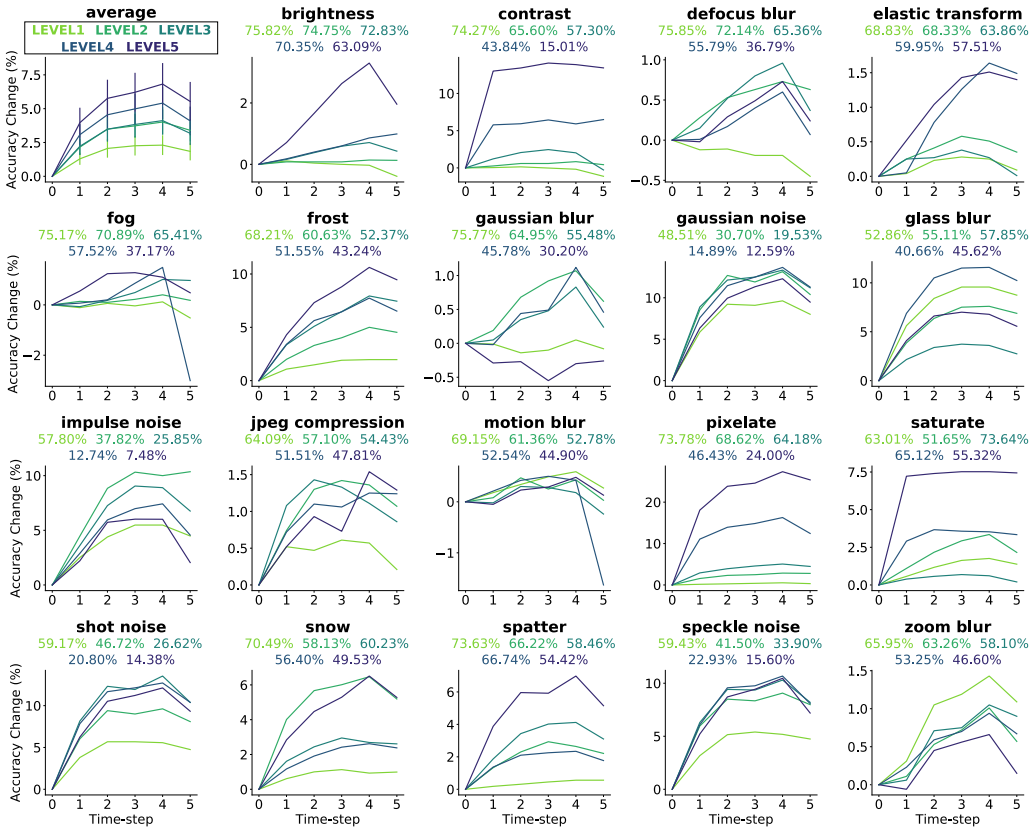

Figure 8: Recognition accuracy of PResNet18 with separate hyper-parameters per PCoder on each of the CIFAR100-C noise types and levels. Each plot shows the change in accuracy with respect to the feedforward baseline (i.e. ResNet18) for each noise type. Each color indicates a noise level. Numbers below the noise names denote the absolute recognition accuracy at time-step 0 (feedforward baseline).

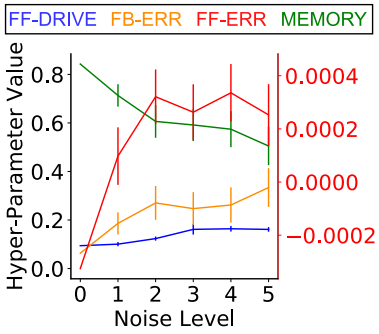

Figure 9: Absolute values of hyper-parameters of PResNet18 when they are shared among PCoders. Each line shows the average value of a hyper-parameter across all 19 CIFAR100-C noise types. Error bars indicate standard error of the mean. The value feedforward error hyper-parameter is plotted with a second y-axis (red). Noise level 0 denotes clean images.

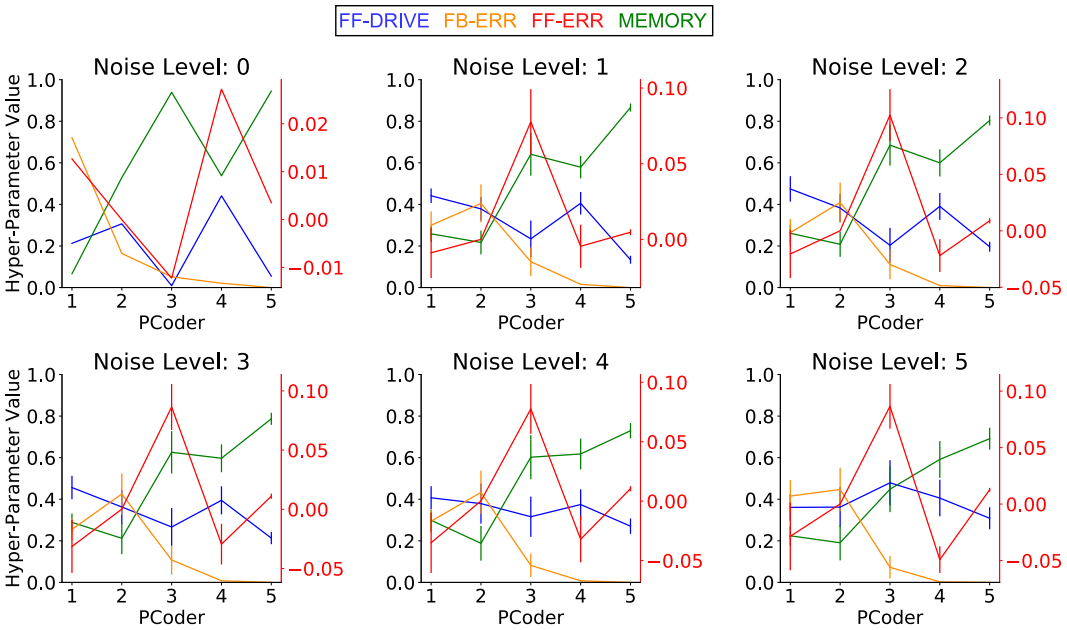

Figure 10: Absolute values of hyper-parameters of PResNet18 when each PCoder uses separate ones. Each plot shows the results of training hyper-parameters as a function of PCoders for a particular noise level. Each line shows the average value of a hyper-parameter across all 19 CIFAR100-C noise types. Error bars indicate standard error of the mean. The value feedforward error hyper-parameter is plotted with a second y-axis (red). Noise level 0 denotes clean images.

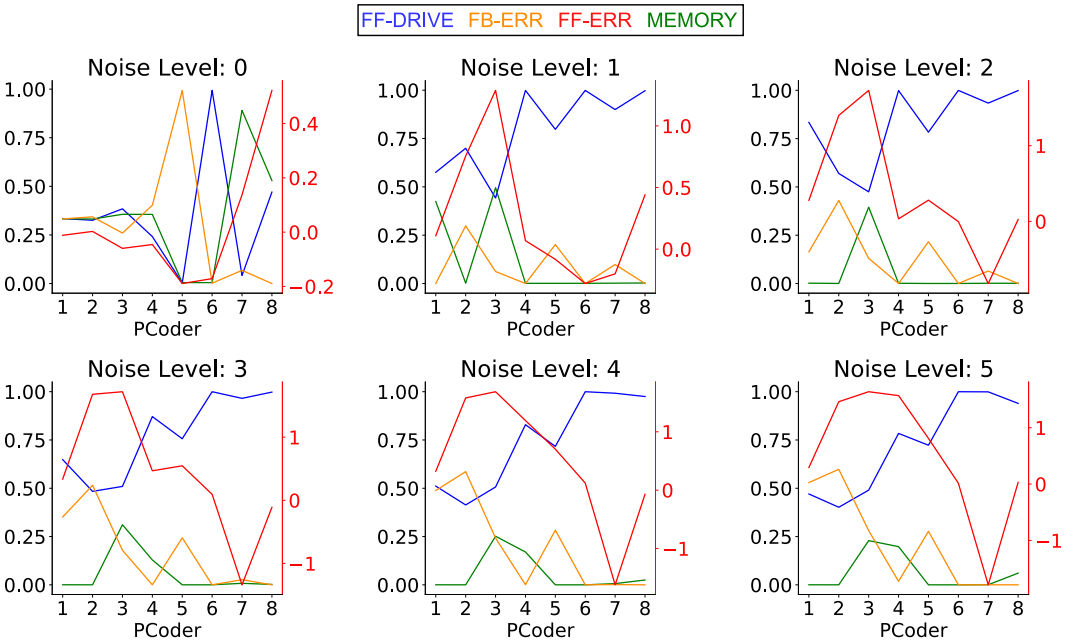

Figure 11: Absolute values of hyper-parameters of PEffNetB0 when each PCoder uses separate ones. Each plot shows the absolute value of a hyper-parameter as a function of PCoders for a particular level of Gaussian noise. The value feedforward error hyper-parameter is plotted with a second y-axis (red). Noise level 0 denotes clean images.

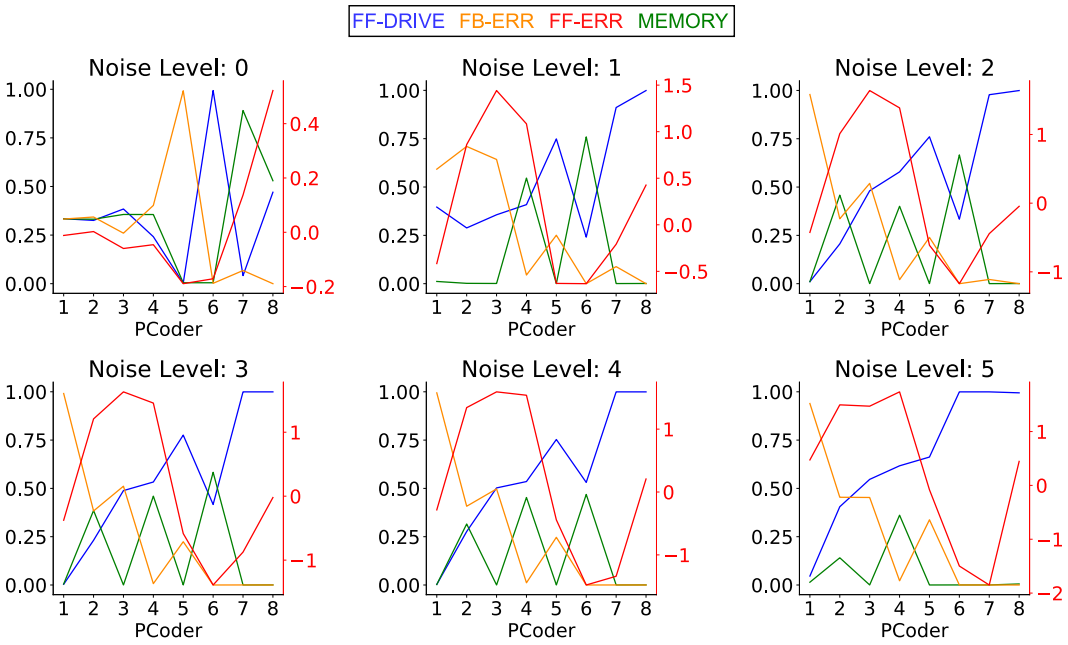

Figure 12: Absolute values of hyper-parameters of PEffNetB0 when each PCoder uses separate ones. Each plot shows the absolute value of a hyper-parameter as a function of PCoders for a particular level of Salt&Pepper noise. The value feedforward error hyper-parameter is plotted with a second y-axis (red). Noise level 0 denotes clean images.