Constrained Parameter Inference as a Principle for Learning

Anonymous authors
Paper under double-blind review

Abstract

Learning in neural networks is often framed as a problem in which targeted error signals are directly propagated to parameters and used to produce updates that induce more optimal network behaviour. Backpropagation of error (BP) is an example of such an approach and has proven to be a highly successful application of stochastic gradient descent to deep neural networks. We propose constrained parameter inference (COPI) as a new principle for learning. The COPI approach assumes that learning can be set up in a manner where parameters infer their own values based upon observations of their local neuron activities. We find that this estimation of network parameters is possible under the constraints of decorrelated neural inputs and top-down perturbations of neural states for credit assignment. We show that the decorrelation required for COPI allows learning at extremely high learning rates, competitive with that of adaptive optimizers, as used by BP. We further demonstrate that COPI affords a new approach to feature analysis and network compression. Finally, we argue that COPI may shed new light on learning in biological networks given the evidence for decorrelation in the brain.

1 Introduction

Learning can be defined as the ability of natural and artificial systems to adapt to changing circumstances based on their experience. In biological and artificial neural networks this requires updating of the parameters that govern the network dynamics (Richards et al., 2019).

A principled way of implementing learning in artificial neural networks is through the backpropagation of error (BP) algorithm (Linnainmaa, 1970; Werbos, 1974). BP is a gradient-based method which uses reverse-mode automatic differentiation to compute the gradients that are needed for individual parameter updating (Baydin et al., 2018). This approach relies on the repeated application of forward and backward passes through the network. In the forward (inference) pass, network activity is propagated forward to compute network outputs. In the backward (learning) pass, the loss gradient associated with the network outputs is propagated in the reverse direction for parameter updating.

While effective, BP makes use of the transmission of gradients using biologically implausible non-local operations, and multiple separated network passes (Grossberg, 1987; Crick, 1989; Lillicrap et al., 2020). Alternative approaches, such as Hebbian learning and subspace methods circumvent this problem yet are restricted to unsupervised learning and do not afford (deep) credit assignment (Brea & Gerstner, 2016; Pehlevan et al., 2015).

Here, we propose constrained parameter inference (COPI) as a new principle for learning. COPI uses information that can be made locally available at the level of individual parameters whose values are being inferred under certain constraints. Note that COPI is distinct from methods that rely on measuring gradients through activity differences (see the NGRAD hypothesis (Lillicrap et al., 2020)), in that no difference in activity needs to be computed to determine parameter updates. Specifically, by constructing a mixed network activity state – in the BP case a simple summation of the forward and backward passes for output units – parameters can infer their own optimal values by observation of node activities alone. This is distinct to many proposed biologically plausible methods which require parameters to measure differences in some activity, either physically using separate compartments/signals, or across time between two phases (Bengio, 2014; Scellier & Bengio, 2017; Ernoult et al., 2020; Whittington & Bogacz, 2017; Sacramento et al., 2018; Payeur et al., 2021). Thus COPI provides a framework which might in future enable online continuous learning where parameter updates are based upon single state measurements.

Furthermore, the COPI algorithm is not tied to any particular credit assignment method. In this sense it assumes that credit can be assigned to units (by a user's method of choice) and simply describes how parameters should update their values given a network state observation. Credit assignment is integrated into COPI by top-down 'perturbations'. The form of the perturbation is precisely what determines which credit-assignment algorithm is being used for learning within the system, whether based on backpropagation, feedback alignment (Lillicrap et al., 2016; Nøkland, 2016), target propagation (Bengio, 2014; Ahmad et al., 2020) or otherwise. Thus, COPI does not address credit assignment as such but rather proposes a general approach for learning based upon a single 'mixed' state regime.

In the following, we demonstrate that COPI provides a powerful framework for learning which is at least as effective as backpropagation of error while having the potential to rely on local operations only. This has direct implications for modern machine learning as COPI can be used as a replacement for the parameter-updating step in backpropagation applications across a wide range of settings.

2 Methods

In this section, we develop the constrained parameter inference (COPI) approach and describe its use for parameter estimation in feedforward neural networks.

2.1 Deep neural networks

Let us consider deep neural networks consisting of L layers. The input-output transformation in layer l is given by

$$y_l = f(a_l) = f(W_l x_l)$$

with output y_l, activation function f, activation $a_l = W_l x_l$, input x_l and weight matrix $W_l \in \mathbb{R}^{K_l \times K_{l-1}}$, where K_l indicates the number of units in layer l. As usual, the input to a layer $l > 1$ is given by the output of the previous layer, that is, $x_l = y_{l-1}$. Learning in deep neural networks amounts to determining for each layer in the network a weight update Δ_{W_l} such that the update rule

$$W_l \leftarrow W_l + \eta \Delta_{W_l}$$

converges towards those weights that minimize a loss ℓ for some dataset $\mathcal{D}$ given a suitable learning rate $\eta > 0$. Locally, the optimum by gradient descent (GD) is to take a step in the direction of the negative expected gradient of the loss. That is,

$$\Delta^{\mathrm{gd}}_{W_l} = -\mathbb{E}\left[\nabla_{W_l} \ell\right] ,$$

where, in practice, the expectation is taken under the empirical distribution.

2.2 Constrained parameter inference in feedforward systems

Here, we develop an alternative approach and relate it directly to both stochastic gradient descent and local parameter inference. Note that the key transformation in a deep feedforward neural network is carried out by a weight matrix given by

$$a_l = W_l x_l \, .$$

Suppose we know the desired target state z_l for this transformation. This can be expressed as an alternative transformation

$$z_l = D_l x_l$$

for some desired weight matrix D_l. Ideally, we would like to use a learning algorithm which guarantees convergence of the current weight matrix to the desired weight matrix. A straightforward proposal is to carry out a decay from the current weight values to the desired weight values, such that the weight update is of the form

$$\Delta_{W_l} = \mathbb{E}\left[D_l - W_l\right] = D_l - W_l \,. \tag{1}$$

Of course, the key goal is to achieve this weight update without making use of the (unknown) desired weight matrix. How to achieve this, is described in the following sections.

2.3 Learning the forward weights

Let us rewrite the desired weight matrix in the following way:

$$\begin{aligned} D_l &= D_l \left(\mathbb{E}\left[x_l x_l^\top\right] \mathbb{E}\left[x_l x_l^\top\right]^{-1} \right) \\ &= \mathbb{E}\left[D_l x_l x_l^\top\right] \mathbb{E}\left[x_l x_l^\top\right]^{-1} \\ &= \mathbb{E}\left[z_l x_l^\top\right] \mathbb{E}\left[x_l x_l^\top\right]^{-1} \end{aligned}$$

with $\mathbb{E}\left[x_l x_l^\top\right]$ the expected (empirical) correlation matrix, which we here assume to be invertible, though this condition is later shown to be unnecessary. If we plug this back into Eq. (1) then we obtain

$$\Delta_{W_l} = \mathbb{E}\left[z_l x_l^\top\right] \mathbb{E}[x_l x_l^\top]^{-1} - W_l \,, \tag{2}$$

allowing the weight update to be expressed in terms of target outputs, z_l, rather than (unknown) desired weights. This is an expression of the least-squares optimisation algorithm.

Let us assume for the moment that the inputs x_l are distributed such that they have zero covariance and unit variance, i.e., the inputs are whitened. This implies that the expected correlation matrix is given by the identity matrix, that is, $\mathbb{E}\left[x_l x_l^\top\right] = I$. In this case, Eq. (2) reduces to the simple update rule

$$\Delta_{W_l} = \mathbb{E}\left[z_l x_l^\top\right] - W_l \,.$$

In practice, it may be unreasonable (and perhaps even undesirable) to assume perfectly whitened input data. A more realistic and achievable scenario is one in which we make the less restrictive assumption that the data is decorrelated rather than whitened. This implies that the expected correlation matrix is diagonal, that is, $\mathbb{E}\left[x_l x_l^\top\right] = \operatorname{diag}\left(\mathbb{E}\left[x_l^2\right]\right)$ with $x_l^2 = x_l \circ x_l$ the vector of squared elements of x with $\circ$ the Hadamard product. Right-multiplying both sides of Eq. (2) by this expression, and assuming that $\mathbb{E}\left[x_l x_l^\top\right]$ is indeed diagonal, we obtain

$$\Delta_{W_l} \operatorname{diag}\left(\mathbb{E}\left[x_l^2\right]\right) = \mathbb{E}\left[z_l x_l^\top\right] - W_l \operatorname{diag}\left(\mathbb{E}\left[x_l^2\right]\right) \,.$$

The matrix multiplication on the right-hand side amounts to a rescaling of the columns of Δ_{W_l}. This is akin to a scaling of the learning rate of the rows of this matrix. This finally leads to our constrained parameter inference (COPI) learning rule

$$\Delta_{W_l}^{\text{copi}} = \mathbb{E}\left[z_l x_l^\top\right] - W_l \operatorname{diag}\left(\mathbb{E}\left[x_l^2\right]\right) \,, \tag{3}$$

which is solely composed of a correlational, Hebbian, learning term and a weight decay term. COPI receives its name from the fact that there are two constraints in play. First, the availability of a target or ‘desired’ state z_l for each layer and, second, the requirement that the inputs x_l are decorrelated.

2.4 Input decorrelation

We did not yet address how to ensure that the inputs to each layer are decorrelated. To this end, we introduce a new decorrelation method which transforms the potentially correlation-rich outputs y_{l-1} of a layer into decorrelated inputs x_l to the following layer using the transformation

$$x_l = R_l y_{l-1},$$

where R_l is a decorrelating ‘lateral’ weight matrix.

Let us consider attempting to reduce the correlation in the output data, x_l, by measurement of its correlation and a shift toward lower correlation. In particular, consider a desired change in a given sample of the form

$$x_l \leftarrow x_l - \eta \left(\mathbb{E}\left[x_l x_l^\top \right] - \operatorname{diag}\left(\mathbb{E}\left[x_l^2 \right] \right) \right) x_l \,,$$

where the expectations could be taken over the empirical distribution. Note that this decorrelating transformation can also be derived rigorously as a gradient descent method, see Appendix E. We can shift this decorrelating transformation from the output activities x_l to the decorrelating matrix R_l. To do so, consider substituting $x_l = R_l y_{l-1}$, such that we may write

$$\begin{aligned} x_l &\leftarrow x_l - \eta \left(\mathbb{E}\left[x_l x_l^\top \right] - \operatorname{diag}\left(\mathbb{E}\left[x_l^2 \right] \right) \right) x_l \\ &\leftarrow R_l y_{l-1} - \eta \left(\mathbb{E}\left[x_l x_l^\top \right] - \operatorname{diag}\left(\mathbb{E}\left[x_l^2 \right] \right) \right) R_l y_{l-1} \\ &\leftarrow (R_l - \eta \left(\mathbb{E}\left[x_l x_l^\top \right] - \operatorname{diag}\left(\mathbb{E}\left[x_l^2 \right] \right) \right) R_l) y_{l-1} \,. \end{aligned}$$

We converge to the optimal decorrelating matrix R_l^* using an update $R_l \leftarrow R_l + \eta \Delta_{R_l}$ with

$$\begin{aligned} \Delta_{R_l}^{\text{copi}} &= -\left(\mathbb{E}\left[x_l x_l^\top \right] - \operatorname{diag}\left(\mathbb{E}\left[x_l^2 \right] \right) \right) R_l \\ &= -\left(\mathbb{E}\left[x_l q_l^\top \right] - \operatorname{diag}\left(\mathbb{E}\left[x_l^2 \right] \right) R_l \right) \end{aligned} \tag{4}$$

with $q_l = R_l x_l$. This update rule bears some similarity to the COPI rule for learning forward weights.

Both the theoretically derived, and biologically plausible formulations of the decorrelating learning rules are measurably more consistent in reducing correlation in the output states, x_l, see Appendix E for a more detailed explanation. Appendix D describes how this method can be written in a more biologically plausible form.

2.5 Error signals

Equation (3) expresses learning of forward weights in terms of target states z_l. However, without access to targets for each layer of a deep neural network model, one may wonder how this learning approach could be applied in the first place. To this end, we assume that the target states can be expressed as

$$z_l = a_l + \alpha \delta_l$$

with δ_l an error signal which perturbs the neuron's state in a desired direction and α a gain term. As described in the following, different error signals can be used to induce effective perturbations.

Gradient-based signals

According to stochastic gradient descent (SGD), the optimal perturbation is given by

$$\delta_l^{\text{sgd}} = -\frac{d\ell}{da_l}$$

as this guarantees that the neuronal states are driven in the direction of locally-decreasing loss. The error signal at the output layer is given by

$$\delta_L \triangleq \delta_L^{\text{sgd}} = -\frac{\partial \ell}{\partial a_L} = -\operatorname{diag}(f'(a_L)) \frac{\partial \ell}{\partial y_L} \,.$$

Starting from $\delta_L^{\text{bp}} = \delta_L$, the SGD perturbation can be computed via backpropagation (BP) by propagating the error signal from output to input according to

$$\delta_l^{\text{bp}} = \frac{\partial a_{l+1}}{\partial a_l} \delta_{l+1}^{\text{bp}}$$

with

$$\begin{aligned}\frac{\partial a_{l+1}}{\partial a_l} &= \frac{\partial y_l}{\partial a_l}\frac{\partial x_{l+1}}{\partial y_l}\frac{\partial a_{l+1}}{\partial x_{l+1}}\\ &= \frac{\partial f(a_l)}{\partial a_l}\frac{\partial R_{l+1}y_l}{\partial y_l}\frac{\partial W_{l+1}x_{l+1}}{\partial x_{l+1}}\\ &= \operatorname{diag}(f'(a_l))R_{l+1}^\top W_{l+1}^\top\,.\end{aligned}$$

While this provides a gold standard for the optimal perturbation, ideally, we would like to replace this by more biologically-plausible credit assignment methods that do not make use of gradient information. These methods typically use the same error signal δ_L for the output layer but propagate the error signal in the input direction using different proposal mechanisms.

Feedback alignment

Feedback alignment (FA) supposes that the perturbation from the previous layer can be propagated through fixed random top-down weights B_{l+1} instead of the transposed weight matrix $(W_{l+1}R_{l+1})^\top$ (Lillicrap et al., 2016). Hence, the layer-wise perturbations are propagated by

$$\delta_l^{\text{fa}} = \operatorname{diag}(f'(a_l))B_{l+1}\delta_{l+1}^{\text{fa}}\,.$$

Direct feedback alignment instead makes use of a random projection of the output node errors to the rest of the network (Nøkland, 2016). That is, we could construct layer-wise perturbations where

$$\delta_l^{\text{dfa}} = \operatorname{diag}(f'(a_l))B_{l+1}\delta_L^{\text{dfa}}\,.$$

Target propagation

Target propagation (TP) supposes that credit is assigned by a top-down, autoencoder-like, pass from the target output toward the input (Bengio, 2014). Supposing a top-down (approximate) inverse model, using weight matrices B_{l+1}, exists for every layer l, where $f\left(B_{l+1}y_{l+1}\right) \approx y_l$, this can be integrated with COPI using a perturbation

$$\delta_l^{\text{tp}} = f\left(B_{l+1}\delta_{l+1}^{\text{tp}}\right) - a_l\,,$$

which indicates a perturbation away from the current activation, a_l, and toward the inverse model $f(B_{l+1}\delta_{l+1}^{\text{tp}})$ of the perturbation from the layer above. Target propagation is not directly related to backpropagation but a more direct theoretical relationship could also be established via methods such as gradient-adjusted incremental target propagation (Ahmad et al., 2020).

In our analyses, we will restrict ourselves to comparing error signals provided by backpropagation and feedback alignment only. Note, however, that the other credit assignment methods can be seamlessly integrated in our setup if desired.

2.6 Stochastic COPI

Stochastic COPI replaces the expectations over the empirical distributions in Eqs. (3) and (4) by single data points, analogous to stochastic gradient descent (SGD). COPI training on single data points proceeds by computing the stochastic weight updates. For all COPI implementations, the forward weight updates are given by

$$\Delta_{W_l}^{\text{copi}} = z_l x_l^\top - W_l \operatorname{diag}\left(x_l^2\right),$$

with target states $z_l = a_l + \alpha\delta_l$ given some suitable error signal δ_l. The decorrelating 'lateral' weight updates are given by

$$\Delta_{R_l}^{\text{copi}} = -\left(x_l q_l^\top - \operatorname{diag}\left(x_l^2\right)R_l\right)$$

with $q_l = R_l x_l$. See Appendix A for a pseudo-algorithm. In practice, as usual, we train on minibatches instead of individual data points.

Correspondence to stochastic gradient descent

For comparison against SGD, it is instructive to consider (stochastic) COPI in the case of a single-layer neural network which is acting on decorrelated inputs. Recall that the SGD update of a single-layer network is given by

$$\Delta_W^{\text{sgd}} = -\frac{d\ell}{da} x^\top .$$

We can manipulate this expression in order to relate SGD to COPI as follows:

$$\begin{aligned} \Delta_W^{\text{sgd}} &= -\frac{d\ell}{da} x^\top \\ &= \left(a - \frac{d\ell}{da} \right) x^\top - a x^\top \\ &= \left(a + \delta^{\text{sgd}} \right) x^\top - W \left(x x^\top \right) . \end{aligned}$$

Comparing this final expression of the SGD update, to the (stochastic) COPI update, namely $\Delta_W^{\text{copi}} = (a+\delta)x^\top - W \operatorname{diag}\left(x^2\right)$, clearly, in this special case, the SGD update and the COPI update with error signal δ^{sgd} are approximately equivalent. The key difference here is that COPI specifically assumes that there are decorrelated inputs and therefore the weight decay term is unaffected by sample-wise input cross-correlations, whereas SGD's weight decay is modified by sample-wise input cross-correlations which could eventually average out to a diagonal matrix in the case of decorrelated inputs.

2.7 Experimental details

Code, pseudocode, implementation details, and additional mathematical explanations of the methods used to produce all plots in the results which follow are provided in the Appendices and Supplementary Material.

3 Results

In the following, we analyse both the convergence properties of COPI and the benefits of the decorrelated representations at every network layer.

3.1 COPI performance on standard benchmarks

To validate COPI as a principle for learning, we compared it against backpropagation by training fully-connected deep feedforward neural networks trained on the MNIST handwritten digit dataset (LeCun et al., 2010) and the CIFAR-10 image dataset (Krizhevsky & Hinton, 2009).

COPI was simulated together with both BP-based (δ_l^{bp}) and FA-based (δ_l^{fa}) perturbations with a loss function composed of the quadratic loss between network outputs and the one-hot encoded labels as 'desired' output. To clarify the benefits of the COPI decorrelation process, we additionally trained networks in which the forward weights W_l were updated by BP and lateral weights R_l were introduced and trained by the COPI decorrelating algorithm, labelled 'BP (with decorrelation)'. Furthermore, we obtained baselines with backpropagation alone (no decorrelation) combined with the Adam optimizer (Kingma & Ba, 2014). Figure 1 shows the results of these simulations. The network architectures, pseudocode, and parameters are described in Appendix A and B.

Figure 1 shows that methods incorporating the COPI decorrelation rule are extremely effective, achieving higher accuracies and lower losses during training, than even a momentum-based approach (Adam). Benefits are observed not only when the decorrelation rule is combined with COPI but also when combined with vanilla SGD (BP). We observe that COPI outperforms BP when both are making use of the decorrelation rule, with COPI achieving significantly higher accuracies and significantly lower losses. We can only attribute this benefit to the explicit difference in the forward COPI and BP rules, where COPI has the built-in assumption that the inputs have a decorrelated form. In terms of test performance, we see in both Figure 1A and 1B that COPI performs on par, often even marginally exceeding the performance of other baselines.

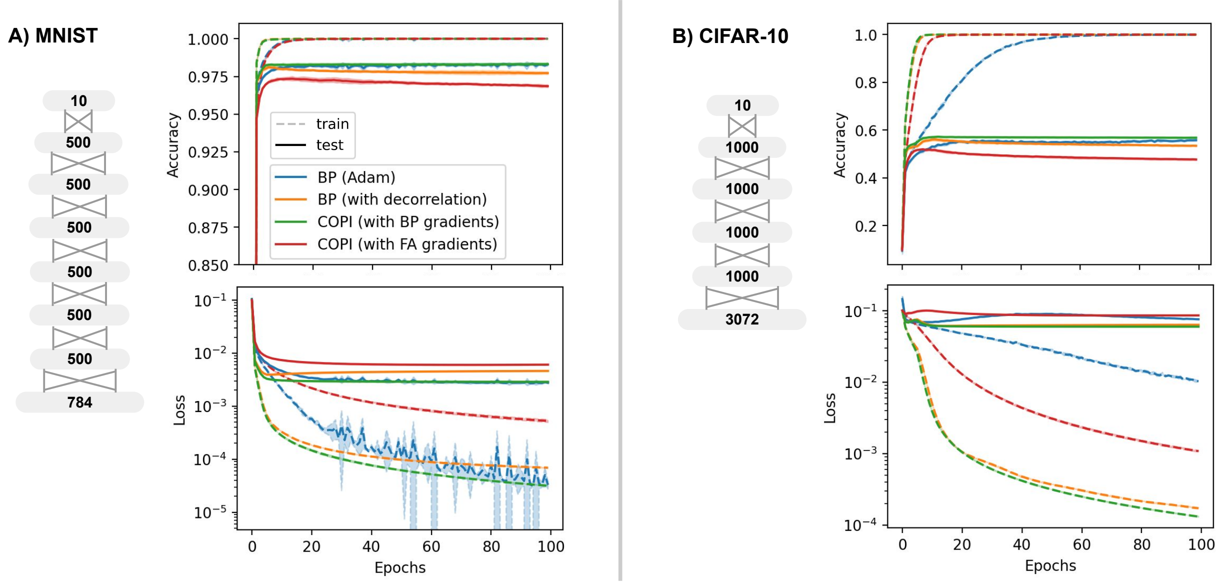

Figure 1: **COPI vs BP performance on standard computer vision classification tasks**. A) Train/test accuracy and loss of a 7-layer (see graphical depiction), fully-connected, feedforward deep neural network model trained and tested on the handwritten MNIST dataset. B) Train/test accuracy of a five-layer, fully-connected, feedforward deep neural network model trained and tested on the CIFAR-10 dataset. All networks were run with five random seeds and envelopes show standard deviation across these networks.

Table 1: **Peak performance (accuracy) measures of COPI vs BP for the results presented in Figure 1**. Also provided in brackets is the mean epoch at which the networks reached 99% of this peak performance.

Method	Peak Performance ± Standard Dev. (Mean # Epochs to 99% of Peak)			
	MNIST		CIFAR-10	
	train	test	train	test
BP (Adam)	$1.0 \pm 0.0\,(6)$	$\mathbf{0.9838 \pm 0.0004\,(5)}$	$0.9998 \pm 0.0001\,(53)$	$0.5619 \pm 0.0023\,(36)$
BP (with decorrelation)	$1.0 \pm 0.0\,(3)$	$0.9812 \pm 0.0009\,(3)$	$1.0 \pm 0.0\,(8)$	$0.5616 \pm 0.0047\,(8)$
COPI (BP gradients)	$1.0 \pm 0.0\,(3)$	$0.9834 \pm 0.0007\,(3)$	$1.0 \pm 0.0\,(7)$	$\mathbf{0.5729 \pm 0.0016\,(10)}$
COPI (FA gradients)	$1.0 \pm 0.0\,(7)$	$0.9740 \pm 0.0010\,(4)$	$1.0 \pm 0.0\,(13)$	$0.5207 \pm 0.0022\,(6.0)$

In general, when compared to backpropagation with a momentum-based optimiser (Adam), we find that the decorrelation coupled networks (both COPI and BP) are able to learn in a much faster manner, reaching maximum training accuracy. This difference in convergence speed, as well as peak performance values are presented in Table 1. As can be seen, models employing decorrelation reach close to peak performance (within 99% of peak performance) much more rapidly even than models trained with the Adam optimiser. Furthermore, the accuracy of COPI trained networks is not compromised, either in training or testing.

3.2 Decorrelation for feature analysis and network compression

The COPI algorithm's requirement for decorrelation at every network layer is not only a restriction but also proves beneficial in a number of ways. We explore the decorrelation, as well as the analyses and computations that it enables.

The proposed decorrelation method produces a representation similar to existing whitening methods, such as ZCA (Bell & Sejnowski, 1997). Figure 2A provides a visualisation of a randomly selected set of data samples

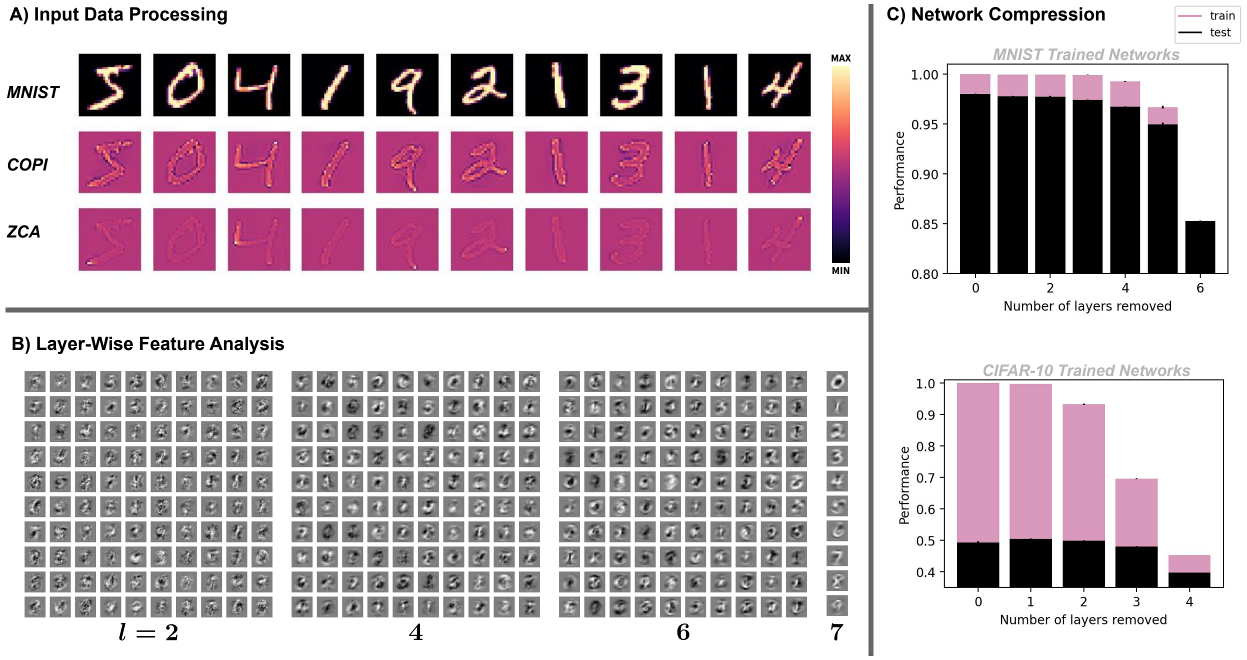

Figure 2: **The effect and utility of decorrelation within COPI network layers for feature readout and network compression**. A) Visualisation of the MNIST dataset (top), the decorrelating transformation produced by the COPI algorithm (middle), and, the whitening transformation produced by ZCA whitening (bottom). B) Using the decorrelated network inputs of the COPI networks (see middle row in A), the decorrelated inputs, x_0, from the entire training set data could be (pixel-wise) correlated with the network's layer-wise outputs, a_l (node-wise). This produced a linear approximation of the units' preferred features from different network layers. C) The decorrelated layer-wise inputs of COPI-trained networks could also be used to efficiently infer linear mappings between distant layers of the network. This allows the removal of layers of a network and replacement with an inferred, linear approximation of those intermediate layers. Plotted are the performances of the 7-layer MNIST trained COPI networks (top) and 5-layer CIFAR-10 trained COPI Networks (bottom) from Figure 1. This network compression process is repeated for five, randomly seeded networks and error bars show standard deviation across these repeats. Layers are removed from the output layer backwards.

from the MNIST dataset. From top to bottom are shown: the unprocessed samples, samples processed by the first decorrelating layer of a network trained on MNIST with the COPI algorithm (see MNIST networks described in Figure 1A), and finally a visualisation of samples transformed by a ZCA transform computed on the whole training set. As can be seen, there is qualitative similarity between the COPI- and ZCA-processed data samples. Remaining differences are attributed to the fact that COPI does not scale the individual elements of these samples (pixels) for unit variance, i.e. whitening, but instead produces decorrelation alone in a distributed fashion.

Beyond the input transformation, COPI allows for visualisation of features deeper in the network by exploiting decorrelated layer-wise inputs. That is, we may use the decorrelated input training dataset to form a linear approximation of the receptive field of units deep in the network. For each decorrelated input unit, we can correlate its activity across the entire training set with the corresponding activity of any unit in the network. This provides an average, linear, feature response for each unit of a given layer.

Figure 2B shows such extracted features from a random selection of 100 units from the second, fourth, sixth and seventh layer of a COPI-trained network (ten for the final layer, which only holds 10 units). These results are from a single network used to produce the results shown in Figure 2A. This allows us to observe an increasingly digit-oriented feature preference for units in our COPI-trained network, as we go deeper into the network, and the presence of digit selective and 'anti-digit' selective units.

This same mechanism of producing a linear approximation of receptive fields also provides a computationally simple method to approximate the transformation produced by multiple layers of a COPI-trained network with a linear matrix. That is, we may employ the COPI principle to infer a linear matrix approximating the transformation across multiple network layers. Appendix F provides an explanation of how this approximation can be made and why this is particularly efficient in networks with decorrelated activities. This approach allowed us to rapidly convert MNIST and CIFAR-10 trained networks into networks consisting of any smaller number of layers, effectively providing a straightforward approach for network compression.

Figure 2C shows the performance impact of such conversions for our seven-layer trained networks of Figure 1A (MNIST trained), and our five-layer trained networks of Figure 1B (CIFAR-10). Note that this approximation is done in a single step using the network's response to the training data and does not require any retraining. Given this, the network performance stays relatively high despite the approximation and removal of layers. In fact, for CIFAR-10, we even see that this approximation returns some small gain in test-set performance. Note that layers are removed sequentially from the end of the network and, as can be seen, there is a significant drop in performance when the first layer of the network is approximated, indicating that the transformations in this layer are crucial for achieving high performance levels.

4 Discussion

In this paper, we introduced constrained parameter inference as a new approach to learning in feedforward neural networks. We derived an effective local learning rule and showed that, under the right conditions, individual weights can infer their own values. The locality required the removal of confounding influences between unit activities within every layer of the neural network, and to this end, we derived an efficient decorrelation rule. We further assumed error signals were available to perturb unit activations towards more desirable states from which the system could learn.

The resulting algorithm allowed us to effectively train deep feedforward neural networks, where performance is competitive with that of backpropagation for both gradient-based and feedback alignment signals. Furthermore, our setup enables much higher effective learning rates than are possible than with vanilla BP and thus allows us to learn at speeds exceeding those possible even using adaptive optimizers. This may contribute to reducing carbon footprint when training large network models (Strubell et al., 2019). The algorithm also allows for more interpretable deep learning via the visualisation of deep decorrelated features (Rudin, 2019; Ras et al., 2022) and could contribute to efficient deep learning as it facilitates network compression (Wang, 2021).

Going forward, it is important to expand the tasks to which COPI is applied and investigate its application to a broader class of network architectures. For example, COPI is in principle compatible with other network components such as convolutional layers, but requires careful consideration as for how to carry out decorrelation in an optimal manner.

From a theoretical standpoint, COPI relates to unsupervised methods for subspace learning (Oja, 1982; Földiák & Young, 1998; Pehlevan et al., 2015). In particular, the form of the learning rule we propose bears a resemblance to Oja's rule (Oja, 1982), though it focuses on inference of parameters in the face of perturbations instead of latent factor extraction. See Appendix G for a comparison.

Aside from unsupervised methods, the inference of parameters based upon input and output activities has been previously proposed to overcome the weight-transport problem (Akrout et al., 2019; Ahmad et al., 2021; Guerguiev et al., 2019). In particular, these methods attempt to learn the feedback connectivity required for backpropagation via random stimulation of units and a process of weight inference. Our method similarly attempts to carry out weight inference, but does so without random stimulation and with the purpose of learning of the forward model through combination with top-down perturbations.

It is also interesting to note that our decorrelating mechanism captures some of the key elements of batch normalization (Ioffe & Szegedy, 2015; Huang et al., 2018). First, vanilla batch-normalization makes use of demeaning, a natural outcome of our decorrelation. Furthermore, whitening of batches has been recently shown to be an extremely effective batch-wise processing stage, yielding state-of-the-art performance on a number of challenging classification tasks (Huang et al., 2018), and reduction of covariance between hidden unit activities has been found to be a generalisation encouraging regularizer (Cogswell et al., 2015). However, unlike all of these methods, our method is not batch-computed and is instead a fixed component of the network architecture, learned over the course of the whole dataset and integrated as a network component.

COPI may also shed light on learning and information processing in biological systems. There is both experimental and theoretical evidence that input decorrelation is a feature of neural processing through a number of mechanisms including inhibition, tuning curves, attention, and eye movements (Franke et al., 2017; Bell & Sejnowski, 1997; Pitkow & Meister, 2012; Segal et al., 2015; Vogels et al., 2011; Abbasi-Asl et al., 2016; Cohen & Maunsell, 2009; Dodds et al., 2019; Graham et al., 2006). In particular, center-surround filters of the LGN appear to produce a form of whitening. Whitening also appears to be key for sparse coding of visual inputs (King et al., 2013). To what extent there is decorrelation between all units projecting to a neuron is of course questionable, though COPI has the potential for modification to account for global or local correlations. In Appendix D, we suggest how the decorrelation rule here might operate in a local fashion.

Beyond this, inhibitory and excitatory balance (Denève & Machens, 2016) has been formulated in a fashion which can be viewed as encouraging decorrelation. Learning rules which capture excitatory/inhibitory balance, such as the one by Vogels et al. (2011), rely on correlative inhibition between units, which in turn reduce the inter-unit covariance. Such detailed balance has been observed across cortical areas and so it does not seem unreasonable to consider this as a method to encourage decorrelation of not just the input but also downstream ‘hidden’ layers of neural circuits.

When considering biological plausibility, the current work assumes that error signals are available and do not interfere with ongoing network activity. This means that we rely on a two-phase credit assignment process. For a fully online implementation, the error machinery should be integrated into a single mixed pass, which is an area for future exploration.

We conclude that constrained parameter inference allows for efficient and effective training of deep feedforward neural networks while also providing a promising route towards biologically plausible deep learning.

References

Reza Abbasi-Asl, Cengiz Pehlevan, Bin Yu, and Dmitri Chklovskii. Do retinal ganglion cells project natural scenes to their principal subspace and whiten them? In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pp. 1641–1645, November 2016.

Nasir Ahmad, Marcel A J van Gerven, and Luca Ambrogioni. GAIT-prop: A biologically plausible learning rule derived from backpropagation of error. *Advances in Neural Information Processing Systems*, 33, December 2020.

Nasir Ahmad, Luca Ambrogioni, and Marcel A J van Gerven. Overcoming the weight transport problem via spike-timing-dependent weight inference. *Neurons, Behavior, Data Analysis, and Theory*, 5(3):1–20, August 2021.

Mohamed Akrout, Collin Wilson, Peter C Humphreys, Timothy Lillicrap, and Douglas Tweed. Deep learning without weight transport. *ArXiv*, April 2019.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, January 2018.

Anthony J Bell and Terrence J Sejnowski. The “independent components” of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, December 1997.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *ArXiv*, July 2014.

Johanni Brea and Wulfram Gerstner. Does computational neuroscience need new synaptic learning paradigms? *Current Opinion in Behavioral Sciences*, 11:61–66, October 2016.

Michael Cogswell, Faruk Ahmed, Ross B Girshick, C L Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *International Conference on Learning Representations*, May 2015.

Marlene R Cohen and John H R Maunsell. Attention improves performance primarily by reducing interneuronal correlations. *Nature Neuroscience*, 12(12):1594–1600, December 2009.

Francis Crick. The recent excitement about neural networks. *Nature*, 337:129–132, 1989.

Sophie Denève and Christian K Machens. Efficient codes and balanced networks. *Nature Neuroscience*, 19 (3):375–382, March 2016.

Eric Mcvoy Dodds, Jesse Alexander Livezey, and Michael Robert DeWeese. Spatial whitening in the retina may be necessary for V1 to learn a sparse representation of natural scenes. *BioRxiv*, pp. 776799, September 2019.

Maxence Ernoult, Julie Grollier, Damien Querlioz, Yoshua Bengio, and Benjamin Scellier. Equilibrium propagation with continual weight updates. *ArXiv*, April 2020.

Peter Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64(2): 165–170, December 1990.

Peter Földiák and Malcolm P Young. Sparse coding in the primate cortex. In *The Handbook of Brain Theory and Neural Networks*. MIT Press, October 1998.

Katrin Franke, Philipp Berens, Timm Schubert, Matthias Bethge, Thomas Euler, and Tom Baden. Inhibition decorrelates visual feature representations in the inner retina. *Nature*, 542(7642):439–444, February 2017.

Daniel J Graham, Damon M Chandler, and David J Field. Can the theory of “whitening” explain the center-surround properties of retinal ganglion cell receptive fields? *Vision Research*, 46(18):2901–2913, September 2006.

Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, January 1987. ISSN 0364-0213.

Jordan Guerguiev, Konrad P Kording, and Blake A Richards. Spike-based causal inference for weight alignment. *ArXiv*, October 2019.

Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 791–800, April 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, pp. 1–11, 2015.

Paul D King, Joel Zylberberg, and Michael R DeWeese. Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1. *Journal of Neuroscience*, 33(13):5475–5485, March 2013.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv*, December 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, July 2009.

Yann LeCun, Corinna Cortes, and Christopher J C Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, November 2016.

Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, June 2020.

Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pp. 6–7, 1970.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in Neural Information Processing Systems*, 29, December 2016.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, November 1982.

Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, 24(7): 1010–1019, July 2021.

Cengiz Pehlevan, Tao Hu, and Dmitri B Chklovskii. A Hebbian/anti-Hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *ArXiv*, March 2015.

Xaq Pitkow and Markus Meister. Decorrelation and efficient coding by retinal ganglion cells. *Nature Neuroscience*, 15(4):628–635, March 2012.

Varun Ranganathan and Alex Lewandowski. ZORB: A derivative-free backpropagation algorithm for neural networks. *ArXiv*, November 2020.

Gabrielle Ras, Ning Xie, Marcel A J van Gerven, and Derek Doran. Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research*, 73:329–397, January 2022.

Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W Lindsay, Kenneth D Miller, Richard Naud, Christopher C Pack, Panayiota Poirazi, Pieter Roelfsema, João Sacramento, Andrew Saxe, Benjamin Scellier, Anna C Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P Kording. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, October 2019.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019.

João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in Neural Information Processing Systems*, 31: 8721–8732, December 2018.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24, May 2017.

Irina Yonit Segal, Chen Giladi, Michael Gedalin, Michele Rucci, Mor Ben-Tov, Yam Kushinsky, Alik Mokeichev, and Ronen Segev. Decorrelation of retinal response to natural scenes by fixational eye movements. *Proceedings of the National Academy of Sciences U.S.A.*, 112(10):3110–3115, March 2015.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *ArXiv*, 2019.

T P Vogels, H Sprekeler, F Zenke, C Clopath, and W Gerstner. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science*, 334(6062):1569–1573, December 2011.

Shiqiang Wang. Efficient deep learning. *Nature Computational Science*, 1(3):181–182, March 2021. ISSN 26628457.

P Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, Cambridge, MA, 1974.

James C R Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, 29(5):1229–1262, May 2017.

A COPI pseudocode

Algorithm 1 demonstrates COPI training of a deep feedforward neural network using a quadratic loss. For illustration, we use backpropagation-based perturbations δ_l^{bp}.

Algorithm 1 Constrained Parameter Inference

```
procedure COPI(network, data)
▷ network consists of randomly initialized forward weight matrices $W_l$ and lateral weight matrices $R_l$ with $1 \leq l \leq L$
▷ data consists of $N$ input-output pairs $(y_0, y^*)$
▷ Parameters: learning rates $\eta_R$ and $\eta_W$; gain term $\alpha$; number of epochs; batch size
    for each epoch do
        for each batch $= \{(y_0, y^*)\} \subset$ data do
            ▷ Forward pass
            for layer $l$ from $1$ to $L$ do
                $x_l = R_l y_{l-1}$                                          ▷ Decorrelate the input data
                $a_l = W_l x_l$                                              ▷ Compute activation
                $y_l = f(a_l)$                                               ▷ Compute output
            end for
            $\ell = ||y_L - y^*||^2$                                         ▷ Compute loss
            ▷ Backward pass
            for layer $l$ from $L$ to $1$ do
                $\delta_l = -\frac{d\ell}{da_L}$ if $l = L$ else $\delta_l = \frac{da_{l+1}}{da_l}\delta_{l+1}$    ▷ Compute learning signal
            end for
            ▷ Update Parameters (mix passes)
            for layer $l$ from $L$ to $1$ do
                $W_l \leftarrow W_l + \eta_W \left((a_l + \alpha\delta_l)x_l^\top - W_l \operatorname{diag}\left(x_l^2\right)\right)$    ▷ Update forward weights
                $R_l \leftarrow R_l - \eta_R \left(x_l (R_l x_l)^\top - \operatorname{diag}\left(x_l^2\right) R_l\right)$    ▷ Update lateral weights
            end for
        end for
    end for
return network
end procedure
```

B Learning setup and parameters

Note that for all simulations which made use of decorrelation (all COPI networks and BP with decorrelation), the networks were first trained for a single epoch with only the decorrelation rule active. This allowed the network to reach a decorrelated state (the desired state for this learning) before forward weights updating was started.

All networks were constructed with the leaky rectified linear unit (leaky-ReLU) activation function. Training was carried out in mini-batches of size 50 for all simulations (stochastic updates computed within these minibatches are averaged during application). The network transformations are described in the methods above, network execution via pseudocode in Appendix A, and learning rate parameters are described below.

Table 2: **Parameters for CIFAR-10 and MNIST trained networks (cf. Figure 1).**

Parameter	BP (Adam)	COPI (with BP/FA gradients) / BP (with decorr)
LeakyReLU negative slope	0.1	0.1
Learning rate η	0.0001	0.0001
Target booster α	1.0	1000.0
Adam param β_1	0.9	-
Adam param β_2	0.999	-
Adam param ϵ	10^{-8}	-

C COPI with categorical cross-entropy loss

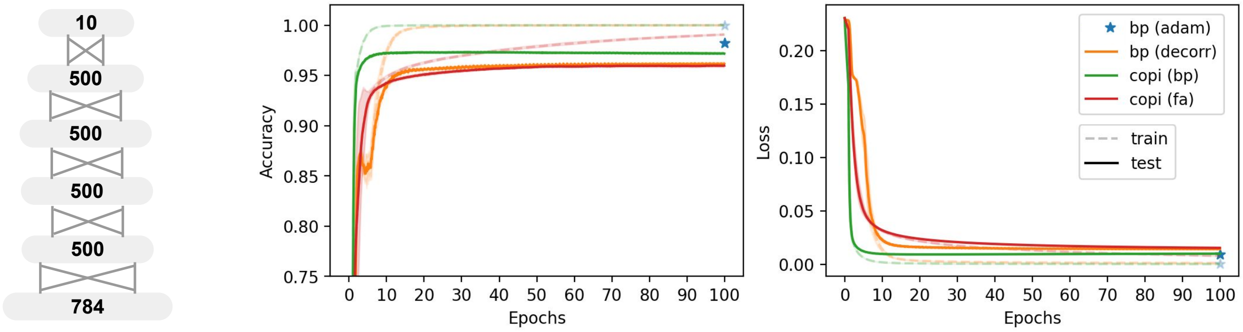

Figure 3: **The performance, measured by training and test accuracy/loss, of a 5-layer fully-connected feedforward neural network architecture trained using a categorical cross-entropy loss**. Plotted are various learning approaches combining decorrelation, our algorithm (copi), and standard stochastic gradient descent by backpropagation of error (bp). All results are shown for training on the MNIST handwritten digit classification task. All networks were run with five random seeds and envelopes show standard deviation across these networks.

In the main text, we explored a quadratic loss in the mathematical derivation and simulation. However, this does not imply that COPI can only be applied in the application of the quadratic loss function, in fact any arbitrary loss function applied to the outputs can be used to produce a gradient-based target.

In particular, take any arbitrary loss which is solely a function of the outputs of a network, $\ell = f(y_L)$. By default, we would compute the derivative with respect to the outputs of the network as $\frac{d\ell}{dy_L}$. This arbitrary formulation differs from a quadratic loss with a target, t_L, since a quadratic loss is proportional to $y_L - t_L$. However, it is possible to reformulate an arbitrary loss function computed on the outputs in terms of a target in the following manner

$$
\begin{aligned}
\frac{d\ell}{dy_L} &= \frac{d\ell}{dy_L} + (y_L - y_L) \\
&= y_L - \left(y_L - \frac{d\ell}{dy_L}\right) \\
&= y_L - t_L^*
\end{aligned}
$$

where t_L^* is a target formulated for this layer.

We make use of such a target-formulation approach (though in terms of the derivative of the output hidden state a_L) in order to train the same network architecture used to train the networks of Figure 1A with a categorical cross-entropy loss. This shows a very successful training approach as previously shown for the COPI networks in the main text, though these simulations have not been as thoroughly optimised by parameter search.

D Biologically plausible decorrelation

The COPI learning rule can also be made more local and biologically plausible by an approximation, which we explore here. In order to make the information locally available for the update of the decorrelation, we make a single assumption: that this correlation reduction can be carried out in either direction – with the target of correlation reduction and source being exchangeable. This assumption allows us to arrive at a more biologically plausible learning rule given by

$$\Delta_{R_l}^{\text{bio-copi}} = -\left(\mathbb{E}\left[q_l x_l^\top\right] - R_l \operatorname{diag}\left(\mathbb{E}\left[x_l^2\right]\right)\right)$$

with $q_l = R_l x_l$. This has exactly the same form as our previous COPI decorrelation rule for learning the forward weights though now acting to update its weights in the same manner as the COPI forward learning rule - using target states q_l. These target states are now the total amount of decorrelating signal being provided to a given unit. Thus this information is available to the post-synaptic unit and could be used for updating. In effect, the lateral weights are also constantly inferring correlational structure within the activities of a layer of units but, given the negatively-signed update, they update their values to reduce correlation instead.

E COPI decorrelation as gradient descent

In the main text, we provided a description of the custom learning rule for decorrelation which forms a part of the COPI learning approach. Here we expand upon this description and frame the same derivation in terms of gradient descent upon a specific loss function.

The COPI learning algorithms require a decorrelated input, meaning that our decorrelation method should minimise the off-diagonal values of $\mathbb{E}[xx^T]$, where x represents the (vector) input data to any given layer and the expectation is taken empirically over a whole dataset. To this end, we can define an element-wise, indexed i, quadratic loss function (l_i), representing the total undesirable correlation induced by a unit, indexed i, with respect to all other units, indexed j, within a single sample such that:

$$l_i = \frac{1}{2} \sum_{j\,:\,j \neq i} \left(x_i x_j\right)^2 .$$

The derivative of this expression can then be taken with respect to unit i, in order to identify how to modify the activity of unit x_i in order to reduce this loss. Specifically,

$$\frac{\partial l_i}{\partial x_i} = \sum_{j\,:\,j \neq i} \left(x_i x_j\right) x_j ,$$

showing that, via stochastic gradient descent, we can produce greater decorrelation by computing the product between unit activities and removing a unit-activity proportional measure from each unit, $x_i \leftarrow x_i - \eta \frac{\partial l_i}{\partial x_i}$, where η would be a learning rate. Vectorising this stochastic gradient descent across all units allows us to write an update for our data x such that

$$x \leftarrow x - \eta \left(xx^T - \operatorname{diag}(x^2)\right) x ,$$

where η is a learning rate and $\operatorname{diag}(\cdot)$, as used in the main text, indicates constructing a square matrix of zeros with the given values upon the diagonal. Finally, as in the main text, we can assume that x

is constructed from some transformation, $x = Ry$, such that we can recast this update in terms of the decorrelating transformation matrix, R, where

$$Ry \leftarrow Ry - \eta \left(xx^T - \text{diag}(x^2)\right) Ry \quad \Rightarrow \quad Ry \leftarrow \left[R - \eta \left(xx^T - \text{diag}(x^2)\right) R\right] y \,,$$

providing an equivalent to our derived update rule for decorrelation $\Delta_R^{\text{copi}} = -\eta \left(xx^T - \text{diag}(x^2)\right) R$, as introduced in the main text.

One may ask why we constructed the specific decorrelation rule described above, rather than using an alternative existing rule. For that matter, one may ask why we chose to take the derivative of our decorrelation loss with respect to unit activities, x, when deriving this rule instead of directly with respect to the parameters of the transformation matrix, R.

First, the used derivation allowed the production of a simple, Hebbian-like update and allowed us to formulate, admittedly by approximation, similar and elegant learning rules for forward and lateral weight matrices. This was important as a promising start in order to work toward methods for online and local learning of these transformations.

Second, on a more rigorous note, the learning rule we propose produces reductions in inter-unit correlations which are not affected by the scale (eigenvalues) of the matrix R. This is a property that is induced by our choice of taking the derivative of our decorrelation loss with respect to the unit activities, x, rather than the matrix elements of R. Note that we can take the derivative of our above loss with respect to a single element of our decorrelating matrix R_{ij} in the following manner,

$$\frac{\partial l_i}{\partial R_{ij}} = \frac{\partial l_i}{\partial x_i} \frac{dx_i}{dR_{ij}} = \sum_{j\,:\, j \neq i} \left(x_i x_j\right) x_j y_j \,.$$

However, reducing correlations by taking the full derivative with respect to the elements of R, or via alternative existing methods which have been proposed for decorrelation through simple anti-Hebbian learning Földiák (1990); Pehlevan et al. (2015), result in a reduction in correlation which is affected by the scale of the matrix R.

We can show this effect empirically in set of simple simulations measuring the magnitude of correlation reduction induced by various learning rules, see Figure 4. In order to produce these results, we first construct a dataset by randomly sampling from a 100-dimensional multivariate normal distribution with a randomly sampled (arbitrary) covariance matrix. We further initialised a matrix $R \in \mathbb{R}^{100\times100}$ composed of the identity function, I, plus random noise added to every element drawn from a [-0.1,0.1] uniform distribution. This matrix, R, is used to process the input data, y, in order to attempt to produce a decorrelated output $x = Ry$ as in the methods of the main part of this paper. In order to simulate an alternative scaling of the matrix R without affecting the output data distribution, we simulate a rescaling of R by removing a constant factor from the input data and scaling R by this factor, $x = (cR)(y/c)$ With this setup, we could then demonstrate how various methods for learning the matrix R (with various scalings applied) reduce the loss function,

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \ell^{(n)} = \frac{1}{N} \sum_{n=1}^{N} \left(x^{(n)} \left(x^{(n)}\right)^{\top} - \text{diag}\left(\left(x^{(n)}\right)^2 \right) \right)^2 ,$$

where n indexes the N samples in the empirical dataset. In Figure 4, the COPI learning rule for decorrelation is compared to the derivative of this loss with respect to the elements of matrix R, $\partial l_i / \partial R_{ij}$ above, and also against a simple anti-Hebbian learning rule Földiák (1990); Pehlevan et al. (2015), where $\Delta_R^{\text{anti-hebbian}} = -(xx^T - \text{diag}(x^2))$. As can be seen, the propose COPI learning rule is the only decorrelating learning rule which reduces the loss function by a consistent amount given some output distribution for x, regardless of the relative scaling of the decorrelating matrix R and the input data y.

Having such a decorrelation method, free from a learning rate interference through the scale of matrix R or the unused pre-decorrelation variable y, is crucial for the COPI learning system since the forward learning rule and decorrelating learning rules interact and must be balanced in their speed of learning to avoid runaway correlations affecting the efficacy of the forward learning rule.

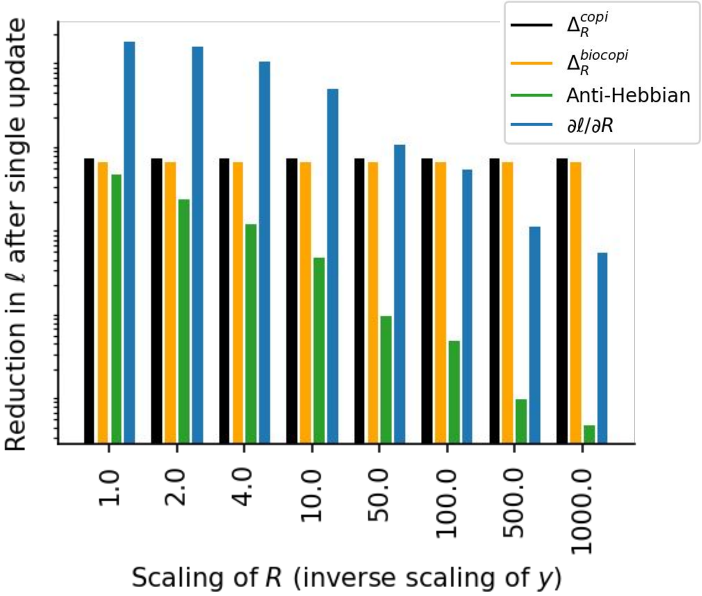

Figure 4: **The reduction in correlation in a a toy-dataset when leveraging various decorrelation rules.** To produce this plot, a dataset was randomly sampled from a multi-variant gaussian distribution and an initial decorrelating matrix R also sampled. This data, with samples y, is processed by the decorrelating matrix R to form outputs, x. A set of methods were then used to compute a single update to the decorrelating matrix, R, and the magnitude of reduction in the loss function (mean of loss $\ell = ||xx^\top - \text{diag}(x^2)||_2^2$ over all datapoints in the dataset) was computed and plotted here. Various scalings were then applied to the matrix R and input data y, which maintained the output data such that $x = (cR)(y/c)$, and the process of computing the efficacy of different learning rules repeated. As shown, only the proposed COPI decorrelating method produces a consistent reduction in the loss function, regardless of the scale of matrix R or the input data y. It is for this reason that this rule is desirable when applied in conjunction with other learning rules which require a relative scaling. The y-scale of this plot is omitted since its scale is arbitrarily dependent upon the initial sampling of data, and this plot is intended to be illustrative.

F Compressing COPI networks

Here we describe how the presence of decorrelated inputs at every network layer enables the compression of a network. Specifically, we make use of these decorrelated inputs in order to approximate multiple (non-linear) layer transformations with a linear matrix.

Consider a system in which one has access to some empirical dataset, $X \in \mathbb{R}^{M\times N}$, where M is the number of features and N the number of datapoints. Suppose also, that there exists some (potentially non-linear) transformation of this dataset, such that $Z = f(X) \in \mathbb{R}^{P\times N}$, where $f(\cdot)$ is some arbitrary function and P is the dimensionality of the transformed data. We could then suppose that we can compute a linear approximation of the function $f(\cdot)$, such that $Z \approx WX$. Supposing that we take this approximation for granted, we might then attempt to solve for the matrix W by

$$Z = WX \Rightarrow W = ZX^{-1} \,.$$

This would only be possible if X was invertible and square. Naturally, a pseudo-inverse could also be deployed, though this is expensive to compute. For such an approach to learning in deep networks, see Ranganathan & Lewandowski (2020). Alternatively, we could (as carried out within the derivation of COPI) multiply our original formulation by the transpose of our data, such that

$$Z = WX \Rightarrow ZX^\top = WXX^\top \Rightarrow W = ZX^\top \left(XX^\top\right)^{-1} \,,$$

where we have now shifted away from an inverse of the data to, instead, the inverse of its correlation matrix. However, we nonetheless still require an (expensive) computation of a matrix inverse. In the case of networks trained to have decorrelated inputs at every layer, we can choose these inputs as our input data, such that $\left(XX^\top\right)^{-1} = \operatorname{diag}\left(1/x_1x_1^\top, \ldots, 1/x_Mx_M^\top\right) \triangleq D$ for these layer activities with x_m the mth row of X. This allows us to compute a transformation from such decorrelated inputs as

$$W = ZX^\top D\,.$$

Thus, in networks with decorrelated layer-wise inputs or activities, linear approximations of transformations can be computed without computing inverses. This is ultimately the mechanism by which COPI also operates, though in a sample/batch-wise manner with a changing output distribution (due to the learning signals). Specifically, consider the COPI algorithm at its 'fixed point', where $\Delta_{W_l}^{\text{copi}} = 0 = \mathbb{E}\left[z_l x_l^\top\right] - W_l \operatorname{diag}\left(\mathbb{E}\left[x_l^2\right]\right)$. Under this condition, we could re-arrange to say that $W_l = \mathbb{E}\left[z_l x_l^\top\right] \operatorname{diag}\left(\mathbb{E}\left[x_l^2\right]\right)^{-1}$, equivalent to our above formulation but under the assumption that we have a fixed, desired output, z_l, unlike the case where we wish to compute stochastic errors and do online learning.

The MNIST dataset consists of many input pixels with a consistently zero activation across all data samples (in the periphery). Due to this, the removal of hidden layers was carried out by removal of layers following the first network layer and the first network layer was never removed. This ensured that our decorrelated data, X, used for the determination of a linear approximation, had no elements which were zero and thus we could divide by X without issue.

G Single-weight updates

Let us consider the COPI update for single synaptic weights, given by

$$\Delta_{w_{ij}}^{\text{copi}} = z_i x_j - w_{ij} x_j^2 = \left(\sum_j w_{ij} x_j\right) x_j - w_{ij} x_j^2 + \delta_i x_j$$

$$\Delta_{r_{ij}}^{\text{copi}} = -(\tilde{z}_i x_j - r_{ij} x_j^2) = \left(\sum_j (-r_{ij}) x_j\right) x_j - (-r_{ij}) x_j^2\,.$$

The first term in both expressions is a Hebbian update which relies on the states of the pre-synaptic units x_j and post-synaptic units z_i or $\tilde{z}_i$ only. The second term in both expressions takes the form of a weight decay. This functional form is similar to Oja's rule Oja (1982), which states:

$$\Delta_{m_{ij}}^{\text{oja}} = y_i x_j - m_{ij} y_i^2 = \left(\sum_j m_{ij} x_j\right) x_j - m_{ij} y_j^2$$

for $y = Mx$. COPI differs from Oja's rule in that the forward weight update has an additional term $\delta_i x_j$ and the weight decay for both the forward and lateral weights depends on the (squared) pre-synaptic rather than post-synaptic firing rates. The functional impact of the difference in the weight decay scaling (by post- vs pre-synaptic firing rates), is that Oja's rule scales weight decay in order to normalize the scale of the weights which target a post-synaptic neuron. By comparison, COPI scales the weight decay in order to best infer the scale which the weights should take in order to reproduce the post-synaptic neuron activity given the observed pre-synaptic neuron activity.