

---

# Deep Predictive Coding Network for Object Recognition

---

Haiguang Wen<sup>1</sup>, Kuan Han<sup>1</sup>, Junxing Shi<sup>1</sup>, Yizhen Zhang<sup>1</sup>, Eugenio Culurciello<sup>2</sup>, Zhongming Liu<sup>1,2</sup>

## Abstract

Inspired by predictive coding in neuroscience, we designed a bi-directional and recurrent neural net, namely deep *predictive coding networks* (PCN). It uses convolutional layers in both feedforward and feedback networks, and recurrent connections within each layer. Feedback connections from a higher layer carry the prediction of its lower-layer representation; feedforward connections carry the prediction errors to its higher-layer. Given image input, PCN runs recursive cycles of bottom-up and top-down computation to update its internal representations to reduce the difference between bottom-up input and top-down prediction at every layer. After multiple cycles of recursive updating, the representation is used for image classification. In training, the classification error backpropagates across layers and in time. With benchmark data (CIFAR-10/100, SVHN, and MNIST), PCN was found to always outperform its feedforward-only counterpart: a model without any mechanism for recurrent dynamics, and its performance tended to improve given more cycles of computation over time. In short, PCN reuses a single architecture to recursively run bottom-up and top-down process, enabling an increasingly longer cascade of non-linear transformation. For image classification, PCN refines its representation over time towards more accurate and definitive recognition.

## 1. Introduction

Convolutional neural networks (CNN) have achieved great success in image recognition. Classical CNN models, e.g. AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), ResNet (He et al., 2016b), SENets (Hu et al., 2017), NASNet (Zoph et al., 2017), have improved the performance in computer vision, while these models generally become deeper and wider by using more layers (Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2016b) or/and filters (Szegedy et al., 2015; Zagoruyko & Komodakis, 2016). Despite various ways of

architectural reconfiguration, these models all scale up from the same principle of computation: extracting image features by a feedforward pass through stacks of convolutional layers.

Although it is inspired by hierarchical processing in biological visual systems (Hubel & Wiesel, 1968), CNN differs from the brain in many aspects. Unlike CNN, the brain achieves robust visual perception by using feedforward, feedback and recurrent connections (Felleman & Van, 1991; Sporns & Zwi, 2004). Information is processed not only through a bottom-up pathway running from lower to higher visual areas, but also through a top-down pathway running in the opposite direction. Such bi-directional processes enable humans to perform a wide range of visual tasks, including object recognition. For human vision, feedforward processing is essential to rapid recognition (Serre et al., 2007; DiCarlo et al., 2012), e.g. when visual input is too brief to recruit feedback and recurrent processing (Thorpe et al., 1996). However, feedback processing improves object recognition and enables cognitive processes to influence perception (Logothetis & Sheinberg, 1996; Wyatte et al., 2014). In neuroscience, the interplay between feedforward and feedback processes is described by hierarchical *predictive coding* (Rao & Ballard, 1999; Friston & Kiebel, 2009; George & Hawkins, 2009; Bastos et al., 2012; Clark, 2013; Hohwy, 2013). It states that the feedback connections from a higher visual area to a lower visual area carry predictions of lower-level neural activities; feedforward connections carry the errors between the predictions and the actual lower-level activities. As a result, the brain dynamically updates its representations to progressively refine its perceptual and behavioral decisions.

Inspired by this brain theory, we designed a bi-directional and recurrent neural net (i.e. PCN). Given image input to PCN, it runs recursive cycles of bottom-up and top-down computation to update its internal representations towards minimization of the residual error between bottom-up input and top-down prediction at every layer in the network. Using predictive coding as its computational mechanism, PCN differs from feedforward-only CNNs that currently dominate computer vision. It is a model with dynamics that uses recursive and bi-directional computation to extract better representations of the input such that the input is predictable by the extracted representation. When it is unfolded in time, PCN runs a longer cascade of nonlinear transformations by running more cycles of bottom-up and top-down computation

---

<sup>1</sup> Electrical and Computer Engineering, Purdue University.

<sup>2</sup> Biomedical Engineering, Purdue University.

Correspondence to: Zhongming Liu <zmliu@purdue.edu>.

through the same architecture without adding more layers, units, or connections.

To explore its value, we designed PCN with convolutional layers stacked in both feedforward and feedback directions. We trained and tested PCN for image classification with benchmark datasets: CIFAR-10 (Krizhevsky & Hinton, 2009), CIFAR-100 (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), and MNIST (LeCun et al., 1998). Our focus was to explore the intrinsic advantages of PCN over its feedforward-only counterpart: a plain CNN model without feedback connection or any mechanism for recurrent dynamics. It turned out that PCN always outperformed the plain CNN model, and its accuracy tended to improve given more cycles of computation over time. Relative to the classical models, PCN yielded competitive performance in all benchmark tests despite much less layers in PCN. As we did not attempt to optimize the performance by trying many learning parameters or model architectures, there is much room for future studies to further improve or extend the model on the basis of a similar notion.

## 2. Related Work

Current progress in computer vision is more driven by engineering goals as opposed to inspiration from the brain (Rawat & Wang, 2017). Findings from recent studies demonstrate that deep convolutional neural networks use representations similar to those in the brain (Khaligh-Razavi & Kriegeskorte, 2014; Yamins et al., 2014; Güçlü & van Gerven, 2015; Cichy et al., 2016; Eickenberg et al., 2017; Wen et al., 2017). However, many gaps are yet to be filled to bridge biological and artificial visual systems. A biologically plausible model of vision should take into account feedback and recurrent connections, which are abundant in primate brains (Felleman & Van, 1991; Sporns & Zwi, 2004). A limited number of studies have taken on this direction from the perspective of computational neuroscience or computer vision.

O'Reilly et al. demonstrated that feedback connections could enable top-down representations to fill incomplete bottom-up representations to improve recognition of partially occluded objects (O'Reilly et al., 2013). Exploiting a similar idea, Spoerer et al. built a recurrent CNN (with 2 hidden layers) using feedforward, feedback, and lateral connections to enable recurrent processing that dynamically updated the internal representations as the sum of bottom-up, top-down, and lateral contributions (Spoerer et al., 2017). Trained and tested with synthesized images of digits, their recurrent CNN yielded more robust recognition of digits in cluttered and occluded images. However, that model did not embody an explicit computational mechanism to ensure recurrent processing dynamics to converge over time. Although compelling from the neuroscience perspective, the models in the above studies were relatively simple and shallow, and they were not tested in naturalistic visual scenarios of primary interest to computer vision.

In computer vision, Liang et al. added recurrent connections into each layer of a feedforward CNN to allow the activity of each unit to be modulated by activities of its neighboring units within the same layer (Liang & Hu, 2015). Although it was inspired by contextual modulation in biological vision, this model did not account for feedback connections, which are abundant in the brain. Stollenga et al. added feedback connections to a trained CNN to enable attentional selection of filters for the model to achieve better object classification (Stollenga et al., 2014). Recently, Canziani et al. built a bi-directional model with a feedforward discriminant subnet, a feedback generative subnet, as well as lateral connections to bridge the two subnets; training the model for video prediction helped the model yield more stable object recognition given video input (Canziani & Culurciello, 2017). These studies described above highlight the roles of feedback and/or recurrent processes in computing or learning better representations than models with only feedforward processes. What remains unresolved is a biologically plausible mechanism that allows feedforward, feedback, and recurrent processes to interact with one another in order for the model to manifest internal dynamics that support various learning objectives.

In this regard, we may seek inspiration from the brain. Predictive coding is an influential theory of neural processing in vision and beyond (Huang & Rao, 2011; Clark, 2013; Hohwy, 2013) as supported by empirical evidence (Gómez et al., 2014; Bastos et al., 2015; Michalareas et al., 2016; Sedley et al., 2016; van Pelt et al., 2016). In a seminal paper (Rao & Ballard, 1997), Rao and Ballard postulated that the brain learns a hierarchical internal model of the visual world. Each level in this model attempts to predict the responses at its lower level via feedback connections; the error between this prediction and the actual response is sent to the higher level via feedforward connections. Friston et al. further generalized this notion into a unified brain theory for perception and action (Friston, 2008). Chalasani et al. used predictive coding to train a deep neural net to learn a hierarchy of sparse representations of data without supervision (Chalasani & Principe, 2013). Lotter et al. explored video prediction as an unsupervised learning objective based on predictive coding (Lotter et al., 2016); however the model trained in this way may not be able to learn sufficiently abstract representation to support such tasks as object recognition. Spratling et al. explored the use of predictive coding for object recognition; however, their model was limited a shallow network architecture for much simplified scenarios (Spratling, 2017).

Inspired by but different from models in prior studies (Rao & Ballard, 1999; Spratling, 2008, 2017), a hierarchical, bidirectional, and recurrent model is proposed and implemented herein as a brain-inspired model for computer vision. This model operates with the theory of predictive coding to generate dynamic internal representations by recursive bottom-up and top-down computation via feedforward and feedback connections across cascaded layers in a deep hierarchy, and recurrent

connections to convey information over time within the same layer. The internal representations are updated to progressively reduce the error of top-down prediction of lower-level representations, while the prediction errors are conveyed upward to higher levels. To train this network, the representations at the highest level, after multiple cycles of recursive updating, are used to classify the input image. With labeled images, the model parameters are trained through backpropagation in time and across layers. In this study, we trained and tested such a deep predictive coding network (PCN) with several datasets: CIFAR-10, CIFAR-100, SVHN, and MNIST.

### 3. Methods

#### 3.1 Predictive Coding

Central to the theory of predictive coding is that the brain continuously generates top-down predictions of bottom-up inputs. The representation at a higher level predicts the representation at its lower level. The difference between the predicted and actual representation elicits an error of prediction, and propagates to the higher level to update its representation towards improved prediction. This repeats throughout the hierarchy until the errors of prediction diminish, or the bottom-up process no longer conveys any “new” (or unpredicted) information to update the hidden representation. Thus, predictive coding is a computational mechanism for the model to recursively update its internal representations of an image towards convergence.

In the following mathematical description of this dynamic process in PCN, italic lowercase letters are used as symbols for *scalars*, bold lowercase letters for column **vectors**, and bold uppercase letters for **MATRICES**. The representation at layer  $l$  and time  $t$  is denoted as  $\mathbf{r}_l(t)$ . The weights of feedforward connections from layer  $l-1$  to layer  $l$  are denoted as  $\mathbf{W}_{l-1,l}$ . The weights of feedback connections from layer  $l$  to layer  $l-1$  are denoted as  $\mathbf{W}_{l,l-1}$ .

In PCN, the higher-level representation,  $\mathbf{r}_l(t)$ , predicts its lower-level representation as  $\mathbf{p}_{l-1}(t)$  via linear weighting  $\mathbf{W}_{l,l-1}$ , as shown in Eq. (1). The prediction error,  $\mathbf{e}_{l-1}(t)$ , is the difference between  $\mathbf{p}_{l-1}(t)$  and  $\mathbf{r}_{l-1}(t)$  as in Eq. (2).

$$\mathbf{p}_{l-1}(t) = (\mathbf{W}_{l,l-1})^T \mathbf{r}_l(t) \quad (1)$$

$$\mathbf{e}_{l-1}(t) = \mathbf{r}_{l-1}(t) - \mathbf{p}_{l-1}(t) \quad (2)$$

##### 3.1.1 FEEDFORWARD PROCESS

For the feedforward process, the prediction error at layer  $l-1$ ,  $\mathbf{e}_{l-1}(t)$ , propagates to the upper layer  $l$  to update its representation,  $\mathbf{r}_l(t)$ , so the updated representation reduces the prediction error. To minimize  $\mathbf{e}_{l-1}(t)$ , let's define a loss as the sum of the squared errors normalized by the variance of the representation,  $\sigma_{l-1}^2$ , as in Eq. (3).

$$e_{l-1}(t) = \frac{1}{\sigma_{l-1}^2} \|\mathbf{e}_{l-1}(t)\|_2^2 \quad (3)$$

The gradient of  $e_{l-1}(t)$  with respect to  $\mathbf{r}_l(t)$  is as Eq. (4).

$$\frac{\partial e_{l-1}(t)}{\partial \mathbf{r}_l(t)} = -\frac{2}{\sigma_{l-1}^2} \mathbf{W}_{l,l-1} \mathbf{e}_{l-1}(t) \quad (4)$$

To minimize  $e_{l-1}(t)$ ,  $\mathbf{r}_l(t)$  is updated by gradient descent with an updating rate,  $\alpha_l$ , as shown in Eq. (5).

$$\begin{aligned} \mathbf{r}_l(t+1) &= \mathbf{r}_l(t) - \alpha_l \left( \frac{\partial e_{l-1}(t)}{\partial \mathbf{r}_l(t)} \right) \\ &= \mathbf{r}_l(t) + \frac{2\alpha_l}{\sigma_{l-1}^2} \mathbf{W}_{l,l-1} \mathbf{e}_{l-1}(t) \end{aligned} \quad (5)$$

If the weights of feedback connections are the transpose of those of feedforward connections  $\mathbf{W}_{l,l-1} = (\mathbf{W}_{l-1,l})^T$ , the update rule in Eq. (5) can be rewritten as a *feedforward* operation, as in Eq. (6).

$$\mathbf{r}_l(t+1) = \mathbf{r}_l(t) + a_l (\mathbf{W}_{l-1,l})^T \mathbf{e}_{l-1}(t) \quad (6)$$

where the last term indicates forwarding the prediction error from layer  $l-1$  to layer  $l$  to update the representation with an updating rate  $a_l = \frac{2\alpha_l}{\sigma_{l-1}^2}$ .

##### 3.1.2 FEEDBACK PROCESS

For the feedback process, the top-down prediction is used to update the representation at layer  $l$ ,  $\mathbf{r}_l(t)$ , to reduce the prediction error  $\mathbf{e}_l(t)$ . Similar to feedforward process, the error is minimized by gradient descent, where the gradient of  $e_l(t)$  with respect to  $\mathbf{r}_l(t)$  is as Eq. (7), and  $\mathbf{r}_l(t)$  is updated with an updating rate  $\beta_l$  as shown in Eq. (8).

$$\frac{\partial e_l(t)}{\partial \mathbf{r}_l(t)} = \frac{2}{\sigma_l^2} (\mathbf{r}_l(t) - \mathbf{p}_l(t)) \quad (7)$$

$$\begin{aligned} \mathbf{r}_l(t+1) &= \mathbf{r}_l(t) - \beta_l \left( \frac{\partial e_l(t)}{\partial \mathbf{r}_l(t)} \right) \\ &= \left( 1 - \frac{2\beta_l}{\sigma_l^2} \right) \mathbf{r}_l(t) + \frac{2\beta_l}{\sigma_l^2} \mathbf{p}_l(t) \end{aligned} \quad (8)$$

Let  $b_l = \frac{2\beta_l}{\sigma_l^2}$  and Eq. (8) is rewritten as follows.

$$\mathbf{r}_l(t+1) = (1 - b_l) \mathbf{r}_l(t) + b_l \mathbf{p}_l(t) \quad (9)$$

Eq. (9) reflects a *feedback* process that the representation at the higher layer,  $\mathbf{r}_{l+1}(t)$ , generates a top-down prediction,  $\mathbf{p}_l(t)$ , and influences the representation at the lower level,  $\mathbf{r}_l(t)$ .

##### 3.1.3 NONLINEARITY

To add nonlinearity to the above feedforward and feedback processes, a nonlinear activation function is applied to the output of each convolutional layer (except the input layer, i.e.  $l = 0$ ). A rectified linear unit (ReLU) (Nair & Hinton, 2010) converts Eqs. (6) and (9) to nonlinear processes as below.

Nonlinear feedforward process:

$$\mathbf{r}_l(t+1) = \text{ReLU} \left( \mathbf{r}_l(t) + a_l (\mathbf{W}_{l-1,l})^T \mathbf{e}_{l-1}(t) \right) \quad (10)$$

Nonlinear feedback process:

$$\mathbf{r}_l(t+1) = \text{ReLU} \left( (1 - b_l) \mathbf{r}_l(t) + b_l \mathbf{p}_l(t) \right) \quad (11)$$

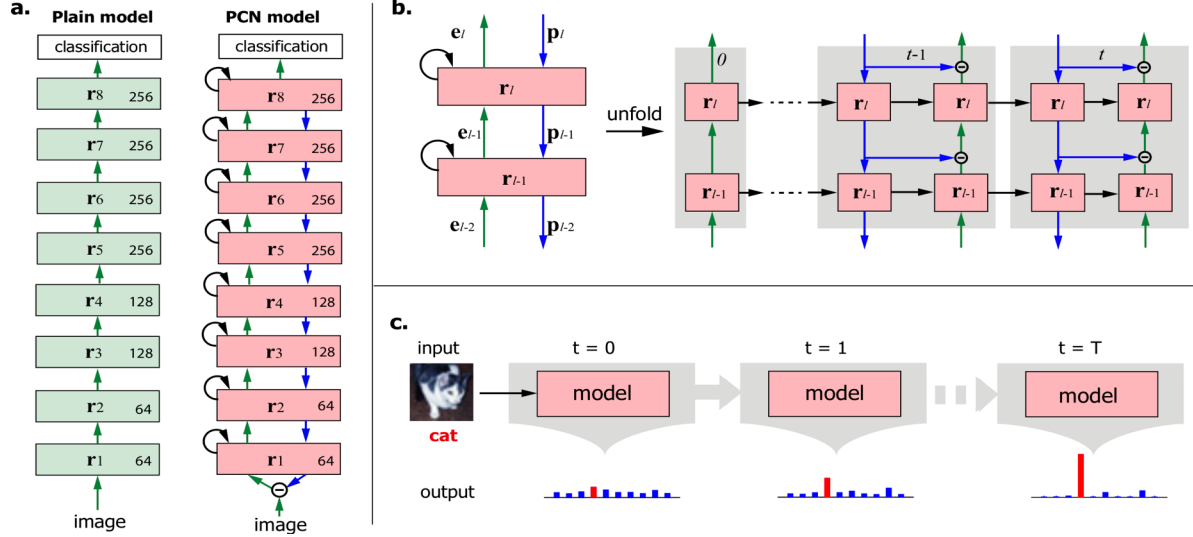


Figure 1. a) An example PCN with 9 layers and its feedforward-only CNN (or the plain model). b) Two-layer substructure of PCN. Feedback (blue), feedforward (green), and recurrent (black) connections convey the top-down prediction, the bottom-up prediction error, and the past information, respectively. c) The dynamic process in the PCN iteratively updates and refines the representation of visual input over time. PCN outputs the probability over candidate categories for object recognition. The bar height indicates the probability and the red indicates the ground truth.

### 3.2 Network Architecture

We implemented this algorithm in several PCNs, all of which included convolutional layers stacked in both feedforward and feedback directions and recurrent connections within each layer as shown in Fig. 1a. These PCNs were trained and tested for object recognition with four benchmark datasets: CIFAR-10, CIFAR-100, SVHN and MNIST. For comparison, several feedforward-only CNNs were built with the same architecture as the feedforward pathway in corresponding PCNs, and were trained and tested with the same datasets. We refer to these feedforward-only CNNs as the plain networks, from which the PCNs were built upon by adding feedback and recurrent connections for dynamic processing.

**Plain CNN Models:** The architecture of our plain CNN models were similar to the architecture of VGG nets (Simonyan and Zisserman, 2014). Briefly, the basic architecture included 6 or 8 convolutional layers and 1 classification layer. All convolutional layers used  $3 \times 3$  filters but different numbers of filters, and used rectified linear unit (ReLU) as the nonlinear activation function. For some layers where the number of filters is doubled, the feature maps were reduced by applying  $2 \times 2$  max-pooling with a stride of 2 after convolution. Batch normalization (Ioffe and Szegedy, 2015) was not used. The classification layer included global average pooling and a fully-connected (FC) layer followed by softmax. On the basis of this architecture, we built 5 VGG-like models that varied in the number of layers and filters, and trained and tested the models with 4 datasets. Table 1 summarizes the architecture of each model.

**Predictive Coding Network (PCN):** Starting from each of the plain CNN models, we added feedback and recurrent connections to form a corresponding PCN. Fig. 1a shows a

9-layer PCN, running recursive bottom-up and top-down processing based on predictive coding. In PCN, feedback connections from one layer to its lower layer were constrained to be the transposed convolution (Dumoulin and Visin, 2016) which is the transpose of the feedforward counterparts, setting apart our models from those in related work on predictive coding (Rao and Ballard, 1999; Spratling, 2008, 2017). As such, both feedforward and feedback connections encoded spatial filters. The former was applied to the errors of the top-down prediction of lower-level representation; the latter was applied to high-level representation in order to predict the lower-level representation. As in the brain, feedforward and feedback connections were reciprocal in PCN. The weights of feedback connections had the identical dimension as the transposed weights of feedforward connections. For layers where max-pooling was applied after feedforward

Table 1. Architectures for PCN. Each column is a model. The layers with the same color have the same feature map size.

CIFAR-10/100			SVHN/ MNIST	
A	B	C	D	E
9 layers	9 layers	7 layers	7 layers	7 layers
input image				
conv3 -64	conv3 -32	conv3 -32	conv3 -32	conv3 -16
conv3 -64	conv3 -32	conv3 -32	conv3 -32	conv3 -16
conv3 -128	conv3 -64	conv3 -64	conv3 -64	conv3 -32
conv3 -128	conv3 -64	conv3 -64	conv3 -64	conv3 -32
conv3 -256	conv3 -128	conv3 -128	conv3 -128	conv3 -64
conv3 -256	conv3 -128	conv3 -128	conv3 -128	conv3 -64
conv3 -256	conv3 -128			
conv3 -256	conv3 -128			
global average pooling, FC-10/100, softmax				

convolution, bilinear unsampling was applied before feedback convolution to ensure that the dimension of top-down prediction could match the dimension of lower-level representation.

An optional constraint to PCN was to use the same set of weights for both feedforward and feedback connections as in some prior studies (Rao and Ballard, 1999; Spratling, 2008, 2017). In other words, the weights of feedback connections were the transposed weights of feedforward connections. With this weight sharing, top-down predictions via feedback connections tended to approach lower-level representations. The PCN would have the same number of parameters as the corresponding plain model. Without this optional constraint of weight sharing, feedforward and feedback weights were assumed to be independent.

---

**Algorithm 1** Deep Predictive Coding Network
 

---

```

Input static image:  $\mathbf{x}$ 
2.  $\mathbf{r}_0(t) \leftarrow \mathbf{x}$ 
3.
4. for  $l = 0$  to  $L-1$  do
5.    $\mathbf{r}_{l+1}(0) \leftarrow \text{ReLU}(\text{FFConv}(\mathbf{r}_l(0)))$ 
6.
7. for  $t = 1$  to  $T$  do
8.   for  $l = L$  to  $1$  do
9.      $\mathbf{p}_{l-1}(t-1) \leftarrow \text{FBConv}(\mathbf{r}_l(t-1))$ 
10.    if  $l > 1$  do
11.       $\mathbf{r}_{l-1}(t-1) \leftarrow \text{ReLU}((1-b)\mathbf{r}_{l-1}(t-1) + b\mathbf{p}_{l-1}(t-1))$ 
12.    for  $l = 0$  to  $L-1$  do
13.       $\mathbf{e}_l(t) \leftarrow \mathbf{r}_l(t) - \mathbf{p}_l(t-1)$ 
14.       $\mathbf{r}_{l+1}(t) \leftarrow \text{ReLU}(\mathbf{r}_{l+1}(t-1) + a\text{FFConv}(\mathbf{e}_l(t)))$ 
15.
16. output  $\mathbf{r}_L(T)$  for classification
    
```

---

**Note:** *FFConv* represents the feedforward convolution, *FBConv* represents the feedback convolution.

### 3.3 Recursive Computation

Unlike feedforward-only networks, PCN runs a dynamic process to update its internal representation throughout the hierarchy (Fig. 1.b). Given an input image, PCN first runs through the feedforward path from the input layer to the last convolutional layer at  $t = 0$ , equivalent to a plain CNN model. For  $t = 1$ , PCN first runs a feedback process and then a feedforward process to update the representations in the hierarchy. In the feedback process, the representation at each layer is updated by a top-down prediction from the higher layer according to Eq. (11). The feedback process runs from the highest convolutional layer to the input layer. In the feedforward process, the representation at each layer is updated by a bottom-up error according to Eq. (10). This procedure is repeated over time. After some cycles, the representation is used as the input to the classification layer to classify the image (see **Algorithm 1**).

### 3.4 Model Training

When PCN is trained for image classification, the error backpropagates across layers and in time to update the model parameters. The update rates are constrained to be non-negative by using ReLU, and are learnable parameters specific to each filter in each layer.

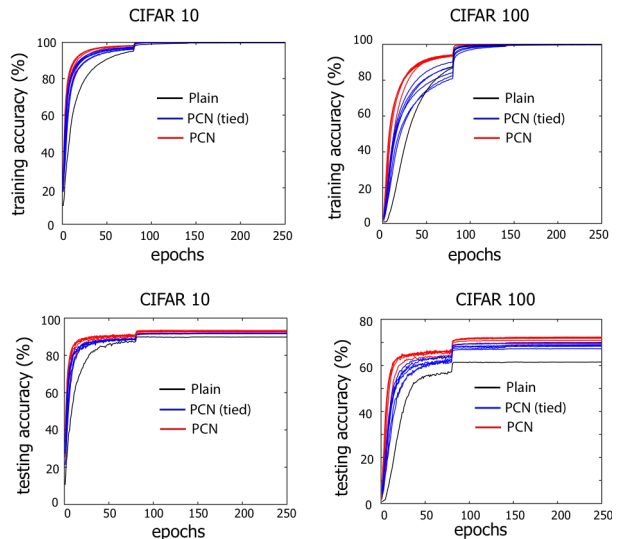
We evaluated two types of PCNs with regard to an optional constraint: the feedforward and feedback connections share the same convolutional weights. With this weight sharing, the feedforward operation and the feedback operation use the same weights. Without the constraint, the feedforward and feedback weights are initialized independently.

In this work, we evaluated these two types of PCNs with a varying number of recursive cycles ( $t = 0, 1, 2, \dots, 6$ ) and with different model architectures (labeled as A through E in Table 1). We use *Plain-A* to represent the plain network with architecture A, and use *PCN-A-t* to represent the PCN with architecture A and  $t$  cycles of recursive computation. *PCN-A-t (tied)* and *PCN-A-t* represent the PCNs *with* and *without* weight sharing, respectively.

We used PyTorch (Paszke et al., 2017) to implement, train, and test the models described above. The convolutional weights and linear weights were initialized to be uniformly random (the default setting in PyTorch). The feedforward and feedback update rates were initialized as 1.0 and 0.5, respectively. The models were trained using mini-batches of a size 128 and without using dropout regularization (Srivastava et al., 2014).

## 4. Experiments

We trained and tested PCN for image classification with data in CIFAR-10/100, SVHN and MNIST, in comparison with plain CNN using the same feedforward architecture. With random initialization, PCN (or CNN) was trained for



**Figure 2.** Training (top) and testing (bottom) accuracies for PCN vs. CNN with matched feedforward architectures for training with CIFAR-10 (left) and CIFAR-100 (right). Each curve represents the average over 5 repeats of one model with different cycles of recursive computation, ranging from 1 to 6.

5 times; the best and mean $\pm$ std top-1 accuracy was reported as below.

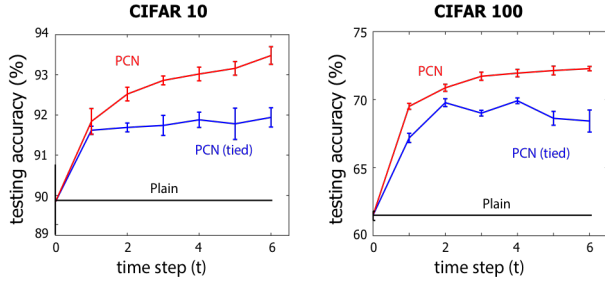


Figure 3. Testing accuracies of PCNs with different time steps.

#### 4.1 CIFAR-10 and CIFAR-100

The CIFAR-10/100 dataset includes 50,000 training images and 10,000 testing images in 10 or 100 object categories. Each image is a  $32 \times 32$  RGB image. PCN (or CNN) were trained on the training set and evaluated on the test set. All images were normalized per channel (i.e. subtract the mean and divide by the standard deviation). For training, we used translation and horizontal flipping for data augmentation. We used stochastic gradient descent to train PCN (or CNN) with a weight decay of 0.0005 and a momentum of 0.9. The learning rate was initialized as 0.01 and was divided by 10 when the error reached the plateau after training for 80, 140, 200 epochs. We stopped after 250 epochs. The hyper-parameters for learning were set

based on validation with 10,000 images in the training set.

##### 4.1.1 PCN vs. CNN

During training, PCN converged much faster than its CNN counterpart (Fig. 2, top), especially when feedforward and feedback connections did not share weights. With testing data, PCN also yielded better accuracy than the plain CNN model (Fig. 2, bottom). For example, *PCN improved* the classification accuracy from 62.11% to 72.48% on CIFAR-100, relative to the plain CNN model. See Table 2 for more results for comparison with other classical or state-of-the-art models. Without being pushed for high accuracy, PCN showed a similar accuracy as ResNet (He et al., 2016b), but relatively lower than the pre-activation ResNet (Pre-act-ResNet) (He et al., 2016a) or the wide residual network (WRN) (Zagoruyko & Komodakis, 2016), which used a much deeper or much wider architecture than the models explored in this study.

##### 4.1.2 PCN WITH DIFFERENT RECURSIVE CYCLES

The accuracy of PCN depended on the number of cycles that recursively updated its internal representations. Fig. 3 shows that the accuracy of PCN tended to increase given more cycles of computation, especially if feedforward and feedback processes did not share the same weights.

To understand why this was the case, we looked into some testing images that were mis-classified by CNN but not by PCN. At each time step (0 through 6), PCN computed a different representation of an image that yielded a different probability distribution across different categories (Fig. 4).

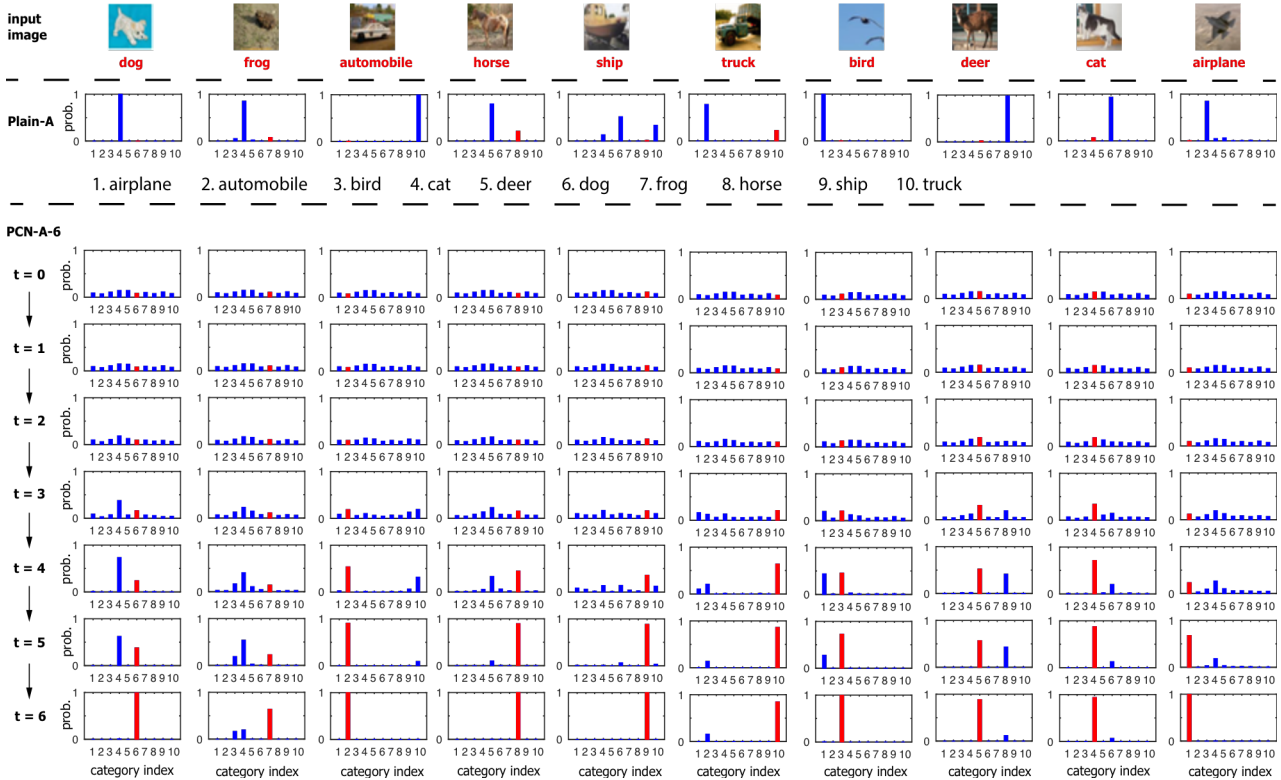


Figure 4. Image classification at different time steps for PCN-A-6 (bottom) in comparison with the plain CNN model (middle) for each of the 10 testing images misclassified by CNN (Plain-A). Each plot shows the probabilities over 10 classes in CIFAR-10. The red represents the ground truth.

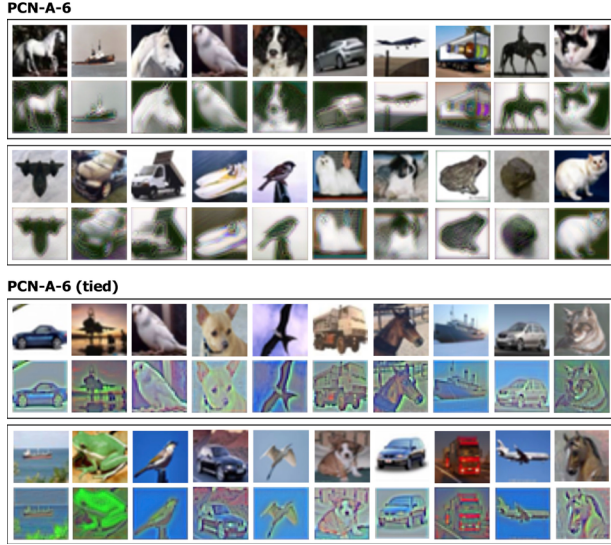


Figure 5. Top-down image prediction by PCN. Here shows example testing images in CIFAR-10 and their corresponding images predicted by PCNs.

Classification was less definitive and/or inaccurate at early time steps. At later time steps, the network corrected itself

Table 2. Compare PCNs with start-of-the-art models on CIFAR-10/100 datasets. #L and #P are the number of layers and parameters, respectively.

Models			CIFAR10/100	
Methods	#L	#P	Accuracy (%)	
Maxout(Goodfellow et al., 2013)	-	-	90.62	61.43
dasNet (Stollenga et al., 2014)	-	-	90.78	66.22
NIN (Lin et al., 2013)	-	-	91.19	64.32
DSN (Lee et al., 2015)	-	-	91.78	65.43
RCNN (Liang & Hu, 2015)	6	1.86M	92.91	68.25
FitNet (Romero et al., 2014)	19	2.5M	91.61	64.96
Highway(Srivastava et al., 2015)	19	2.3M	92.46	67.76
ResNet (He et al., 2016b)	110	1.7M	93.57	-
	164	1.7M	-	74.84
	1001	10.2M	-	72.18
	1202	19.4M	92.07	-
Pre-act-ResNet (He et al., 2016a)	110	1.7M	93.63	-
	164	1.7M	94.54	75.67
	1001	10.2M	95.08	77.29
WRN-40-4	40	8.9M	95.47	78.82
WRN-16-8	16	11M	95.73	79.57
WRN-28-10 (Zagoruyko & Komodakis, 2016)	28	36.5M	96.00	80.75
DenseNet (Huang et al., 2017)	250	15.3M	96.28	82.40
Plain-A	9	2.33M	90.61	62.11
PCN-A-6 (tied)	9	2.33M	92.26	69.44
PCN-A-6	9	4.65M	<b>93.83</b>	<b>72.58</b>
Plain-B	9	0.58M	89.53	62.21
PCN-B-2 (tied)	9	0.58M	90.76	65.57
PCN-B-6	9	1.16M	92.80	69.34
Plain-C	7	0.29M	88.23	61.36
PCN-C-2 (tied)	7	0.29M	89.56	64.09
PCN-C-6	7	0.57M	92.40	68.31

to yield more definitive and accurate classification. It was true especially for ambiguous images, where a cat looked like a dog, or a deer looked like a horse, even for humans. See more examples in Fig. 4.

#### 4.1.3 GENERATIVE PREDICTION IN PCN

When it was trained for image classification, PCN was not explicitly optimized to reconstruct the input image, unlike a previous work that used video prediction as the learning objective (Lotter et al., 2016). Nevertheless, the top-down process in PCN was able to reconstruct the input with high accuracy. Although this was expected for PCN with weight sharing, reconstruction was also reasonable even for PCN without weight sharing (Fig. 5). This result was surprising, and implied that PCN, without any architectural constraint to enable image reconstruction, is able to reshape itself to predict or reconstruct the input, even when it is trained for a discriminative task, e.g. object recognition. Speculatively PCN potentially provides a new way to simultaneously train a discriminative network for object recognition and a generative network for prediction or reconstruction.

## 4.2 SVHN

SVHN is a dataset of Google’s Street View House Numbers images (Netzer et al., 2011) and contains more than 600,000 color images of size  $32 \times 32$ , divided into training set, testing set and an extra set. The task of this dataset is to classify the digit located at the center of each image. Since the task is easier than CIFAR datasets, we implemented PCN with simpler network architectures (see Table 1). To validate the hyper parameters, we randomly selected 400 samples per class from the training set and 200 samples per class from the extra set for validation, as in (Goodfellow et al., 2013). The remainder of the training set and the extra set were used for training. The preprocessing for SVHN was the same as for CIFAR, i.e. per-channel normalization. No data augmentation was used. We used the Adam (Kingma & Ba, 2014) optimization with a weight decay of 0.0005 and an initial learning rate of 0.001 for a 20-10-10 epoch schedule. The exponential decay rates for the first and second moment estimates were 0.9 and 0.99, respectively. Table 3 shows the classification performance for this dataset. Like what we found for the CIFAR dataset, PCN always outperformed the plain CNN counterpart.

## 4.3 MNIST

The MNIST dataset consists of hand written digits 0-9. There are 60,000 training images and 10,000 testing images in total. Each image is a gray image of size  $28 \times 28$ . For this dataset, the same network architecture as used for SVHN is adopted. The training procedure was the same as for SVHN. Table 4 shows the classification performance for this dataset. PCN consistently performed better than its CNN counterpart. The best PCN achieves 0.36% error rate, comparable to some previous state-of-the-art models.

## 5. Discussion and Conclusion

Table 3. Compare PCNs with start-of-the-art models on SVHN. The accuracy was obtained from five repeats.

SVHN			
Methods	#L	#P	error rate (%)
Maxout(Goodfellow et al.,	-	-	2.47
NIN (Lin et al., 2013)	-	-	2.35
Stochastic pooling (Zeiler and Fergus, 2013)	-	-	2.80
Dropconnect (Wan et al.,	-	-	1.94
DSN (Lee et al., 2015)	-	-	1.92
RCNN (Liang and Hu, 2015)	6	2.67M	1.77
FitNet (Romero et al., 2014)	13	1.5M	2.42
WRN-16-8 (Zagoruyko and Komodakis, 2016)	16	11M	1.54
<b>Plain-D</b>	7	0.29M	3.21(3.41±0.13)
<b>PCN-D-2 (tied)</b>	7	0.29M	2.63(2.92±0.11)
<b>PCN-D-6</b>	7	0.57M	2.28(2.42±0.09)
<b>Plain-E</b>	7	0.07M	3.19(3.41±0.13)
<b>PCN-E-1 (tied)</b>	7	0.07M	2.74(2.91±0.11)
<b>PCN-E-6</b>	7	0.14M	<b>2.24(2.42±0.10)</b>

What defines PCN are 1) the use of bi-directional and recurrent connections as opposed to feedforward-only connections, and 2) the use of predictive coding as a mechanism for the model to recursively run bottom-up and top-down processes. When it is trained for image classification, the model dynamically refines its representation of the input image towards more accurate and definitive recognition. As this computation is unfolded in time, PCN reuses a single architecture and the same set of parameters to run an increasingly longer cascade of nonlinear transformation.

We say it is “longer” instead of “deeper”, because the notion behind PCN is different from the mindset in deep learning that more layers are required to model more complex and nonlinear relationships in data. Making a model increasingly deeper is arguably less efficient or scalable, bringing a set of challenges or burdens, e.g. the need for more computational resource and training data. In contrast, the brain does not use a deeper network to do more challenging tasks. A more challenging task simply takes the brain longer time to process information through the same network.

Predictive coding tells PCN how to compute but not how to learn. In this study, PCN is trained for image classification based on the representation emerging from the top layer after multiple cycles of computation. The error of classification backpropagates (top-down and bottom-up) across layers and in time to update the model parameters for multiple times (as many as the cycles of recursive computation) per training example or batch of examples. This helps the learning to converge faster, while utilizing full knowledge in training data. If an image takes the model more cycles of computation to converge its representation, it means that the image has more information than what the model can explain or generate, and thus the image carries a greater value for the model to learn. Therefore, it is more desirable to train PCN for more challenging visual tasks, e.g. images that are ambiguous or

Table 4. Compare PCNs with the start-of-the-art models on MNIST. The accuracy was obtained from five repeats

MNIST			
Methods	#L	#P	error rate (%)
Maxout(Goodfellow et al., 2013)	-	-	0.45
NIN (Lin et al., 2013)	-	-	0.47
Stochastic pooling (Zeiler and Fergus, 2013)	-	-	0.47
Dropconnect (Wan et al., 2013)	-	-	0.21
DSN (Lee et al., 2015)	-	-	0.39
RCNN (Liang and Hu, 2015)	6	0.67M	0.31
FitNet (Romero et al., 2014)	-	-	0.51
Hierarchical PC/BC-DIM (Spratling, 2017)	-	-	2.19
<b>Plain-D</b>	7	0.29M	0.53(0.59±0.04)
<b>PCN-D-1 (tied)</b>	7	0.29M	0.43(0.50±0.06)
<b>PCN-D-1</b>	7	0.57M	0.38(0.46±0.06)
<b>Plain-E</b>	7	0.07M	0.68(0.74±0.03)
<b>PCN-E-1 (tied)</b>	7	0.07M	0.43(0.51±0.06)
<b>PCN-E-4</b>	7	0.14M	<b>0.36(0.48±0.06)</b>

difficult to recognize, while reducing the need for a large number of otherwise “simple” training examples.

For image classification, PCN takes an image as the input for all cycles of its recursive computation, while the errors of top-down prediction sent to the first hidden layer vary across cycles or in time. When the input is not a static image but a video, the input to the first hidden layer represents the errors of prediction of the present video frame given the model’s representations from the past frames. This would enable the model to compute and learn representations of both spatial and temporal information in videos, which is an important aspect that awaits to be explored in future studies.

As an initial step to explore predictive coding in computer vision, it was our intention to start and compare with models with a basic CNN architecture (like that of VGG) in order to focus on evaluation of the value of using predictive coding as a computational mechanism. However, we expect that some network modules are readily applicable to PCN as well as CNN, including batch normalization (Ioffe & Szegedy, 2015) and short-cut connections (He et al., 2016b). In addition, the update rates for top-down and bottom-up computation may be trainable as time-variant parameters as opposed to constants assumed in the current implementation. Augmentation of training data or regularization techniques, e.g. dropout (Srivastava et al., 2014) may also help to improve the model’s performance in image classification. In future studies, we will explore alternative architectures and learning strategies for larger and more training images, e.g. ImageNet (Deng et al., 2009).

## Acknowledgments

The research was supported by NIH R01MH104402 and Purdue University.

## References

- Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., & Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, 76(4), 695-711.
- Bastos, A. M., Vezoli, J., Bosman, C. A., Schoffelen, J.-M., Oostenveld, R., Dowdall, J. R., De Weerd, P., Kennedy, H., & Fries, P. (2015). Visual areas exert feedforward and feedback influences through distinct frequency channels. *Neuron*, 85(2), 390-401.
- Canziani, A., & Culurciello, E. (2017). Cortexnet: a generic network family for robust visual temporal representations. *arXiv preprint arXiv:1706.02735*.
- Chalasani, R., & Principe, J. C. (2013). Deep predictive coding networks. *arXiv preprint arXiv:1301.3541*.
- Cichy, R. M., Khosla, A., Pantazis, D., Torralba, A., & Oliva, A. (2016). Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific reports*, 6.
- Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3), 181-204.
- Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 248-255). IEEE.
- DiCarlo, J. J., Zoccolan, D., & Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, 73(3), 415-434.
- Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.
- Eickenberg, M., Gramfort, A., Varoquaux, G., & Thirion, B. (2017). Seeing it all: Convolutional network layers map the function of the human visual system. *Neuroimage*, 152, 184-194.
- Felleman, D. J., & Van, D. E. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1), 1-47.
- Friston, K. (2008). Hierarchical models in the brain. *PLoS computational biology*, 4(11), e1000211.
- Friston, K., & Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521), 1211-1221.
- George, D., & Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. *PLoS computational biology*, 5(10), e1000532.
- Gómez, C., Lizier, J. T., Schaum, M., Wollstadt, P., Grützner, C., Uhlhaas, P., . . . Hornero, R. (2014). Reduced predictable information in brain signals in autism spectrum disorder. *Frontiers in neuroinformatics*, 8, 9.
- Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.
- Grill-Spector, K., & Kanwisher, N. (2005). Visual recognition: As soon as you know it is there, you know what it is. *Psychological Science*, 16(2), 152-160.
- Güçlü, U., & van Gerven, M. A. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27), 10005-10014.
- He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. (2016a). In *Proceedings of the IEEE Conference on CVPR*, Las Vegas, NV, USA. doi: 10.1109/CVPR.2016.90. 770-778.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision*. 630-645.
- Hohwy, J. (2013). The predictive mind: *Oxford University Press* doi: 10.1093/acprof:oso/9780199682737.001.0001
- Hu, J., Shen, L., & Sun, G. (2017). Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*.
- Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. (Vol. 1, No. 2, p. 3).
- Huang, Y., & Rao, R. P. (2011). Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5), 580-593.
- Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1), 215-243.
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*. 37, 448-456
- Khaligh-Razavi, S.-M., & Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain IT cortical representation. *PLoS Comput Biol*, 10(11), e1003915.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto, 2009*.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in neural information processing systems*. 1097-1105.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015). Deeply-supervised nets. In *the Artificial Intelligence and Statistics*.

- Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on CVPR*. doi: 10.1109/CVPR.2015.7298958
- Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.
- Logothetis, N. K., & Sheinberg, D. L. (1996). Visual object recognition. *Annual review of neuroscience*, 19(1), 577-621.
- Lotter, W., Kreiman, G., & Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*.
- Michalareas, G., Vezoli, J., Van Pelt, S., Schoffelen, J.-M., Kennedy, H., & Fries, P. (2016). Alpha-beta and gamma rhythms subserve feedback and feedforward influences among human visual cortical areas. *Neuron*, 89(2), 384-397.
- Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML*, 807-814.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*.
- O'Reilly, R. C., Wyatte, D., Herd, S., Mingus, B., & Jilk, D. J. (2013). Recurrent processing during object recognition. *Frontiers in psychology*, 4, 124.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Workshop on Autodiff*.
- Rao, R. P., & Ballard, D. H. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural computation*, 9(4), 721-763.
- Rao, R. P., & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2, 79.
- Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9), 2352-2449.
- Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.
- Sedley, W., Gander, P. E., Kumar, S., Kovach, C. K., Oya, H., Kawasaki, H., . . . Griffiths, T. D. (2016). Neural signatures of perceptual inference. *Elife*, 5.
- Serre, T., Oliva, A., & Poggio, T. (2007). A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences*, 104(15), 6424-6429.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Spoerer, C. J., McClure, P., & Kriegeskorte, N. (2017). Recurrent convolutional neural networks: a better model of biological object recognition. *Frontiers in psychology*, 8, 1551.
- Sporns, O., & Zwi, J. D. (2004). The small world of the cerebral cortex. *Neuroinformatics*, 2(2), 145-162.
- Spratling, M. W. (2008). Predictive coding as a model of biased competition in visual attention. *Vision research*, 48(12), 1391-1408.
- Spratling, M. W. (2017). A hierarchical predictive coding model of object recognition in natural images. *Cognitive computation*, 9(2), 151-167.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929-1958.
- Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. In *Proceedings of the Advances in neural information processing systems*. 2, 2377-2385.
- Stollenga, M. F., Masci, J., Gomez, F., & Schmidhuber, J. (2014). Deep networks with internal selective attention through feedback connections. In *Proceedings of the Advances in Neural Information Processing Systems*. 2, 3545-3553.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. doi: 10.1109/CVPR.2015.7298594
- Thorpe, S., Fize, D., & Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, 381(6582), 520.
- van Pelt, S., Heil, L., Kwisthout, J., Ondobaka, S., van Rooij, I., & Bekkering, H. (2016). Beta-and gamma-band activity reflect predictive coding in the processing of causal events. *Social cognitive and affective neuroscience*, 11(6), 973-980.
- Wen, H., Shi, J., Zhang, Y., Lu, K.-H., Cao, J., & Liu, Z. (2017). Neural Encoding and Decoding with Deep Learning for Dynamic Natural Vision. *Cerebral Cortex*, 1-25. doi:10.1093/cercor/bhx268
- Wyatte, D., Jilk, D. J., & O'Reilly, R. C. (2014). Early recurrent feedback facilitates visual object recognition under challenging conditions. *Frontiers in psychology*, 5, 674.
- Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23), 8619-8624.
- Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.
- Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2017). Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*.