Learning long-range spatial dependencies with horizontal gated-recurrent units

Drew Linsley **Junkyung Kim** **Vijay Veerabadran** **Thomas Serre**
Department of Cognitive Linguistic & Psychological Sciences
Carney Institute for Brain Science
Brown University
Providence, RI 02912
{drew_linsley,junkyung_kim,vijay_veerabadran,thomas_serre}@brown.edu

Abstract

Progress in deep learning has spawned great successes in many engineering applications. As a prime example, convolutional neural networks, a type of feedforward neural networks, are now approaching – and sometimes even surpassing – human accuracy on a variety of visual recognition tasks. Here, however, we show that these neural networks and their recent extensions struggle in recognition tasks where co-dependent visual features must be detected over long spatial ranges. We introduce the horizontal gated-recurrent unit (hGRU) to learn intrinsic horizontal connections – both within and across feature columns. We demonstrate that a single hGRU layer matches or outperforms all tested feedforward hierarchical baselines including state-of-the-art architectures which have orders of magnitude more free parameters. We further discuss the biological plausibility of the hGRU in comparison to anatomical data from the visual cortex as well as human behavioral data on a classic contour detection task.

1 Introduction

Consider the images in Fig. 1a: A sample image from the Berkeley Segmentation Data Set (BSDS500) [1] is shown on the left and the corresponding contour map produced by a state-of-the-art deep convolutional neural network (CNN) [2] on the right. Although this task has long been considered challenging because of the need to integrate global contextual information with inherently ambiguous local edge information, CNNs now rival humans at detecting contours in natural scenes [2–6]. Now, consider the image in Fig. 1b, which depicts a variant of a visual psychology task referred to as "Pathfinder" [7]. Reminiscent of the everyday task of reading a subway map to plan a commute (Fig. 1c), the goal in the Pathfinder task is to determine if two white circles in an image are connected by a path. Compared to natural images such as the one shown in Fig. 1a, these images are visually simple, and the task is indeed easy for human observers to solve [7]. Nonetheless, we will demonstrate that modern CNNs struggle to solve this task. Why is it that a CNN can accurately detect contours in a natural scene like Fig. 1a but also struggle to integrate paths in the stimuli shown in Fig. 1b? In principle, the ability of CNNs to learn such long-range spatial dependencies is limited by their localized receptive fields (RFs). This is typically addressed by building deeper networks, which increases the size and complexity of network RFs. Here, we use a large-scale analysis of CNN performance on the Pathfinder challenge to demonstrate that this depth-based solution is inefficient at capturing long-range spatial dependencies.

An alternative solution to problems that stress long-range spatial dependencies is provided by biology. The visual cortex contains abundant horizontal connections which mediate nonlinear interactions between neurons across distal regions of the visual field [8, 9]. These intrinsic connections, popularly called "association fields", are thought to form the main substrate for mechanisms of contour grouping according to Gestalt principles, by mutually exciting colinear elements while also suppressing clutter elements that do not form an extended contour [10–14]. Such "extra-classical receptive field" mechanisms, mediated by horizontal connections, allow receptive fields to adaptively "grow" without additional processing depth. Several computational neuroscience models of these neural circuits have been proposed to account for an array of phenomena from perceptual grouping to contextual

Preprint. Work in progress.

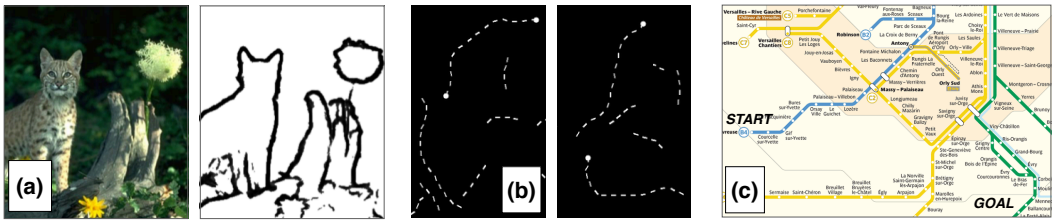

Figure 1: State-of-the-art CNNs excel at natural image contour detection benchmarks, but are strained by a task that depends on detecting long-range spatial dependencies. (a) Representative contour detection performance of a leading model. (b) Exemplars from the Pathfinder challenge: a task consisting of synthetic images which are parametrically controlled for long-range dependencies. (c) Long-range dependencies similar to those in the Pathfinder challenge are critical for everyday behaviors, such as reading a subway map to navigate a city.

illusions [e.g., 10, 15, 16]. However, because these models are fit to data by solving sets of differential equations using numerical integration, they have so far not been amenable to computer vision.

Here, we describe an implementation of these models' core ideas in an end-to-end trainable extension of the popular gated recurrent unit (GRU) [17], which we call the horizontal GRU (hGRU). Unlike CNNs, which exhibit a sharp decrease in accuracy for increasingly long paths, we show that the hGRU is highly effective at solving the Pathfinder challenge with just *one layer* and a fraction of the number of parameters and training samples needed by CNNs. We further show that, when trained on natural scenes, the hGRU learns patterns of horizontal connections that resemble anatomical patterns of horizontal connectivity found in the visual cortex, and exhibits a detection profile that strongly correlates with human behavior on a classic contour detection task [18].

Related work Much previous work on recurrent neural networks (RNNs) has focused on modeling temporal sequences using learnable gates in the form of long-short term memory (LSTM) units [19] or GRUs [17], e.g., for visual recognition in videos [20–23]. This approach was also extended to learning long-range interactions in images by applying RNNs to images transformed into one-dimensional sequences, leading to successful applications such as handwritten character recognition [24], object recognition [25], segmentation [26], image generation, and in-painting [27, 28]

Each of these applications represents the use of recurrent weight matrices with all-to-all spatial connectivity patterns. A more constrained pattern of connectivity is found in the related Conditional Random Fields (CRFs), which were developed explicitly to capture associations between neighboring features. CRFs have been used in image segmentation [29, 30] as a post-processing stage to smooth out and increase the spatial resolution of prediction maps. Originally formulated as probabilistic models, CRFs can also be cast as RNNs [31].

Local connectivity patterns in RNNs have also been implemented through convolutions, which has been adopted as a method to help models achieve greater processing depth for object recognition and super-resolution without introducing additional parameters [32–35]. The current work further builds on this constraint, adding connectivity patterns and circuit mechanisms that are typically found in computational neuroscience models of neural circuits (e.g., [10, 15, 16, 36, 37]).

2 Horizontal gated recurrent units (hGRUs)

GRU We first review the popular gated recurrent unit (GRU) as a method for learning horizontal connections (Fig. 2). The GRU operates on the feedforward drive $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$ from a previous convolutional layer, and is capable, in theory, of encoding spatial dependencies via its (time-varying) hidden state $\mathbf{H}_t \in \mathbb{R}^{H\times W\times C}$. The GRU manages updates to $\mathbf{H}_t$ using two activities, commonly referred to as the input and update "gates": $\mathbf{G}_t^{(1)}, \mathbf{G}_t^{(2)} \in \mathbb{R}^{H\times W\times C}$:

$$\mathbf{G}_t^{(1)} = \sigma(\mathbf{X} + (U^{(1)} * \mathbf{H}_{t-1}) + \mathbf{b}^{(1)}) \tag{1}$$

$$\mathbf{G}_t^{(2)} = \sigma(\mathbf{X} + (U^{(2)} * \mathbf{H}_{t-1}) + \mathbf{b}^{(2)}). \tag{2}$$

The point nonlinearity $\sigma(\cdot)$ transforms activities into $[0, 1]$. Because the transformation this applies is continuous, we more precisely refer to the input and forget gates as the "gain" and "mix". Gain and mix activities are derived from convolutions, denoted by $*$, between the kernels $U^{(1)}, U^{(2)} \in \mathbb{R}^{I\times I\times C}$ with the hidden state $\mathbf{H}_{t-1}$, followed by the addition of the biases $\mathbf{b}^{(1)}, \mathbf{b}^{(2)} \in \mathbb{R}^{I\times I\times C}$.

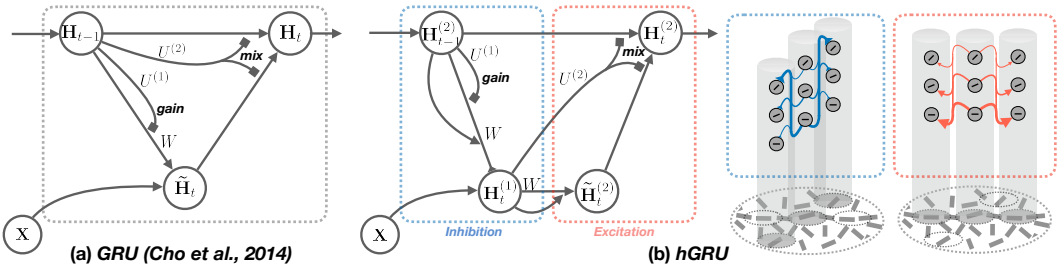

Figure 2: Implementing horizontal connections with **(a)** the GRU and **(b)** the proposed hGRU. Inspired by computational models of neural circuits, the hGRU can learn horizontal connections between units in $\mathbf{X}$ (transparent columns) that implement linear and nonlinear forms of inhibition (blue) and excitation (red).

The gain activity, $\mathbf{G}_t^{(1)}$ is then used to modulate feature channels in $\mathbf{H}_{t-1}$ for computing a candidate hidden state $\tilde{H}_t$:

$$\tilde{H}_t = \zeta(\mathbf{X} + W * (\mathbf{G}_t^{(1)} \odot \mathbf{H}_{t-1}) + \mathbf{b}). \tag{3}$$

In contrast, $\mathbf{G}_t^{(2)}$ selects features of $\tilde{H}_t$ to integrate, or mix, with $\mathbf{H}_{t-1}$ through element-wise multiplication $\odot$. Horizontal connections between units are learned with the kernels $W \in \mathbb{R}^{S \times S \times C}$, where S is the spatial extent of a kernel. The bias $\mathbf{b} \in \mathbb{R}^{1 \times 1 \times C}$ mediates the point nonlinearity $\zeta(\cdot)$. Convolving W with the recurrent state of the network allows it to form long-range inhibitory and excitatory connections between units in $\mathbf{X}$. The hidden state $\mathbf{H}_t$ is then updated as in:

$$\mathbf{H}_t = (\mathbf{1} - \mathbf{G}_t^{(2)}) \odot \mathbf{H}_{t-1} + \mathbf{G}_t^{(2)} \odot \tilde{H}_t \tag{4}$$

The GRU effectively increases the RF of units in $\mathbf{X}$ without introducing pooling or increasing the number of parameters in the model. However, as we will show, the simple linear excitation and inhibition implemented by the GRU is insufficient for the kinds of tasks depicted in Fig. 1.

hGRU We build on the GRU framework to introduce the hGRU: a model with the ability to learn complex forms of inhibition and excitation associated with horizontal connections in the visual cortex. This is done by introducing separate populations of inhibitory and excitatory neurons into the GRU framework. The activity of the populations is calculated in separate stages, which we denote as $\mathbf{H}^{(1)}$ (inhibition) and $\mathbf{H}^{(2)}$ (excitation; see Fig. 2). Like the GRU, this model includes a gain $\mathbf{G}^{(1)}$ and mix $\mathbf{G}^{(2)}$, which modulate the spread of activity across processing time steps. The contributions of horizontal connections in each stage are calculated with the kernels W. Consistent with computational models of neural circuits (e.g., [10, 15, 16, 36, 37]), W is constrained to have symmetric weights between channels. This means that channels indexed by i, j and j, i will have the same strength, which reduces its number of learnable parameters by nearly half compared to a GRU.

We begin by describing the calculation of $\mathbf{H}_t^{(1)}$. The gain $\mathbf{G}^{(1)}$ is applied to $\mathbf{H}_{t-1}^{(2)}$, before convolving the resulting activity with the horizontal kernel W to calculate the contributions from horizontal connections at time t. The resulting activity $\mathbf{C}_t^{(1)}$ is combined with $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ to inhibit $\mathbf{X}$. These parameters are k-dimensional vectors and respectively control linear (subtractive) and nonlinear (divisive, or shunting [10]) inhibition by horizontal connections onto units $\boldsymbol{X}$. The nonlinearities ζ and σ are the same as the GRU. Note that this formulation does not introduce asymmetry between nonlinear vs. linear forms of inhibition [16], which we found was less effective for learning.

$$\mathbf{G}_t^{(1)} = \sigma(\mathbf{X} + (U^{(1)} * \mathbf{H}_{t-1}^{(2)}) + \mathbf{b}^{(1)}) \tag{5}$$

$$\mathbf{C}_t^{(1)} = W * (\mathbf{H}_{t-1}^{(2)} \odot \mathbf{G}_t^{(1)}) \tag{6}$$

$$\mathbf{H}_t^{(1)} = \zeta(\mathbf{X} - (\boldsymbol{\alpha}\mathbf{H}_{t-1}^{(2)} + \boldsymbol{\mu}) \odot \mathbf{C}_t^{(1)})) \tag{7}$$

The updated inhibitory $\mathbf{H}_t^{(1)}$ is next used to calculate the excitatory $\mathbf{H}_t^{(2)}$.

$$\mathbf{G}_t^{(2)} = \sigma(\mathbf{X} + (U^{(2)} * \mathbf{H}_t^{(1)}) + \mathbf{b}^{(2)}) \tag{8}$$

$$\mathbf{C}_t^{(2)} = \boldsymbol{\beta}(W * \mathbf{H}_t^{(1)}) \tag{9}$$

$$\tilde{H}_t^{(2)} = \zeta(\boldsymbol{\kappa}\mathbf{H}_t^{(1)} + \boldsymbol{\kappa}\mathbf{C}_t^{(2)} + \boldsymbol{\omega}\mathbf{H}_t^{(1)} \odot \mathbf{C}_t^{(2)}) \tag{10}$$

$$\mathbf{H}_t^{(2)} = (((\mathbf{1} - \mathbf{G}_t^{(2)}) \odot \mathbf{H}_{t-1}^{(2)}) + \mathbf{G}_t^{(2)} \odot \tilde{H}_t^{(2)})\eta_t \tag{11}$$

The mix $\mathbf{G}^{(2)}$ is calculated as in the GRU. The activity $\mathbf{C}_t^{(2)}$ represents the contribution of horizontal connections, which are a function of the just-calculated $\mathbf{H}_t^{(1)}$. Excitation at this stage is controlled by the k-dimensional parameters $\boldsymbol{\beta}$, $\boldsymbol{\kappa}$, and $\boldsymbol{\omega}$. The parameter $\boldsymbol{\beta}$ is a gain applied to the horizontal-kernel activities $\mathbf{C}_t^{(2)}$, which facilitates learning by freeing the horizontal kernel, which is shared in both stages, from learning to scale excitation through time. The parameters $\boldsymbol{\kappa}$ and $\boldsymbol{\omega}$ control the additive and multiplicative contributions of horizontal connection activities $\mathbf{C}_t^{(2)}$ to $\tilde{H}_t^{(2)}$. With this full suite of excitatory interactions, the hGRU can in principle implement both first- and second-order excitation (i.e., to assess self-similarity), each of which play specific computational roles in perception [38]. Finally, the mix $\mathbf{G}^{(2)}$ integrates the candidate $\tilde{H}_t^{(2)}$ with $\mathbf{H}_t^{(2)}$. To control for disappearing activities associated with long-range spatial dependencies, we introduce the learnable T-dimensional parameter $\boldsymbol{\eta}$, which we refer to as a time-gain. This modulates $\mathbf{H}_t^{(2)}$ with the scalar, $\boldsymbol{\eta}_t$, which as we show in our experiments below improves model performance.

3 The Pathfinder challenge

We evaluated the limits of feedforward and recurrent architectures on the "Pathfinder challenge", a synthetic visual task inspired by cognitive psychology [7]. The task, depicted in Fig. 1b, involves detecting whether two circles are connected by a path. This is made more difficult by allowing target paths to curve and introducing multiple shorter unconnected "distractor" paths. The Pathfinder challenge involves three separate datasets, for which the length of paths and distractors are parametrically increased. This challenge therefore screens models for their effectiveness in detecting complex long-range spatial relationships in cluttered scenes.

3.1 Stimulus design

Pathfinder images were generated by placing oriented "paddles" on a canvas to form dashed paths. Each image contained two paths made of a fixed number of paddles and multiple distractors made of one third as many paddles. Positive examples were generated by placing two circles at the ends of a single path (Fig. 1b, left) and negative examples by placing one circle at the end of each of the paths (Fig. 1b, right). The paths were curved and variably shaped, with the possible number of shapes exponential to the path length. The Pathfinder challenge consisted of three datasets, in which path and distractor length was successively increased, and with them, overall task difficulty. These datasets had path lengths of 6, 9 and 14 paddles, and each contained 1,000,000 unique images of 150×150 pixels. See SI for a detailed description of the stimulus generation procedure.

3.2 Model implementation

We performed a large-scale analysis of the effectiveness of feedforward and recurrent computations on the Pathfinder challenge. We controlled for the effects of idiosyncratic model specifications by using a standard architecture, consisting of "input", "feature extraction", and "readout" processing stages. Swapping different feedforward or recurrent layers into the feature extraction stage let us measure the relative effectiveness of each on the challenge. All models except for state-of-the-art "residual networks" (ResNets) [39] and per-pixel prediction architectures were embedded in this architecture, and these exceptions are detailed below. See SI for a detailed description of the input and readout stage. Models were trained on each Pathfinder challenge dataset (Fig. 3d), with 90% of the images used for training (900,000) and the remainder for testing (100,000). We measured model performance in two ways. First, as the accuracy on test images. Second, as the "area under the learning curve" (ALC), or mean accuracy on the test set evaluated after every 1000 batches of training, which summarized the rate at which a model learned the task. Accuracy and ALC were taken from the model that achieved the highest accuracy across 5 repetitions of training. All models were trained for a single epoch except for the ResNets, which were trained for longer. Model training procedures are detailed in SI.

3.3 Recurrent models

We tested 6 different recurrent layers in the feature extraction stage of the standard architecture: hGRUs with 8, 6, and 4-timesteps of processing; a GRU; and hGRUs with lesions applied to parameters controlling linear or nonlinear horizontal interactions. Both the GRU and lesioned versions of the hGRU ran for 8 timesteps. These layers had 15×15 horizontal connection kernels (W) with an equal number of channels as their input layer (25 channels).

We observed three overarching trends: First, each model's performance monotonically decreased, or "strained", as path length increased. Increasing path length reduced model accuracy (Fig. 3a), and

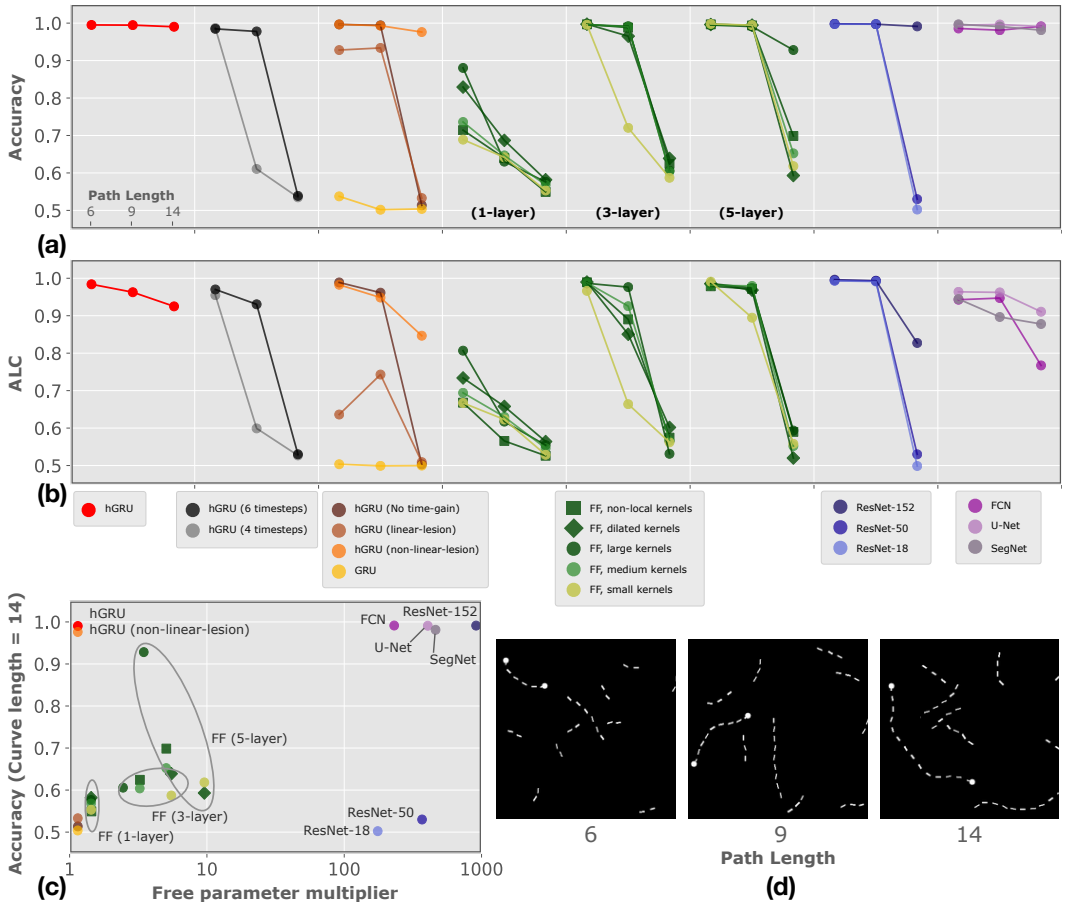

Figure 3: The hGRU efficiently learns long-range spatial dependencies that otherwise strain feedforward architectures. **(a)** Model accuracy is plotted for the three Pathfinder challenge datasets, which featured paths of 6- 9- and 14-paddles. Each panel depicts the accuracy of a different model class after training on the 900,000 images in each dataset. Only the hGRU and state-of-the-art models for classification (the two right-most panels) approached perfect accuracy on each dataset. **(b)** Measuring the area under the learning curve (ALC) of each model (mean accuracy) demonstrates that the rate of learning achieved by the hGRU across the Pathfinder challenge is only rivaled by the U-Net architecture (far right). **(c)** The hGRU is significantly more parameter-efficient than feedforward models at the Pathfinder challenge, with its nearest competitors needing at least 200× the number of parameters to match its performance. The x-axis shows the number of parameters in each model versus the hGRU (as a multiple of the latter). The y-axis depicts model accuracy on the 14-length Pathfinder dataset. **(d)** Pathfinder challenge exemplars of different path lengths (all are positive examples).

increased the number of batches it took to learn a task (Fig. 3b). Second, the 8-timestep hGRU was more effective than any other recurrent model at the challenge, and it outperformed each of its lesioned variants including the standard GRU. Notably, this hGRU was strained the *least* by the Pathfinder challenege out of all tested models, with a negligible drop in accuracy as path length increased. This finding highlights the contributions that each of the hGRU mechanisms make to processing long-range spatial dependencies, in particular emphasizing the importance of the model's mechanisms for nonlinear inhibition and excitation. Third, hGRU performance monotonically decreased with processing time. This revealed a minimum number of timesteps that it needed to solve each Pathfinder dataset: 4 for the length-6 condition, 6 for the length-9 condition, and 8 for the length-14 condition (first vs. second columns in Fig.3a). Such time-dependency in the Pathfinder task is consistent with the accuracy-reaction-time tradeoff found in humans as the distance between endpoints of a curve increases [7].

3.4 Feedforward models

We screened an array of feedforward models on the Pathfinder challenge. Model performance revealed the importance of a kernel size vs. width, model depth, and feedforward operations for incorporating additional scene context for solving the challenge's datasets. Model construction began by embedding the feature extraction stage of the standard model with kernels of one of three different sizes: 10×10, 15×15, or 20×20. These are referred to as small, medium, and large kernel models (Fig. 3). To control for the effect of network capacity on performance the number of kernels given to the each model was varied so that they all had the same number of parameters (36, 16, and 9 kernels; giving each feedforward model at least as many parameters as the hGRU). We also tested two additional feedforward models that featured candidate operations for incorporating contextual information into local convolutional activities. One version used (2-pixel) dilated convolutions, which involves applying a stride to the kernel before convolving the input [40, 41], and has helped performance in multiple computer vision problems including denoising [42] and semantic segmentation [30, 43]. The other version applied a non-local operation to convolutional activities [44], which aids performance in both static and dynamic visual processing tasks. These operations were incorporated into the first feature extraction layer of the medium kernel (15×15 filter) model described above. We also considered deeper versions of each of the above "1-layer" models (referring to the depth of the feature extraction stage), stacking them to build 3- and 5-layer versions. This yielded a total of 15 different feedforward models.

Without exception, the performance of each feedforward model was significantly strained by the Pathfinder challenge. The magnitude of this straining was well predicted by model depth and size, and operations for incorporating additional contextual information made no discernible difference to the overall pattern of results. The 1-layer models were most effective on the 6-length Pathfinder dataset, but were pulled towards chance on the remaining conditions. Increasing model capacity to 3 layers rescued the performance of all but the small kernel model on the 9-length Pathfinder dataset, but even then did little to improve performance on the 14-length dataset. Of the 5-layer models, only the large kernel configuration came close to solving the 14-length dataset. The ALC of this model, however, demonstrates that its rate of learning was slow, especially compared to the hGRU (Fig. 3b). The failures of these feedforward models is all the more striking when considering that each had between $1\times$ and $10\times$ the number of parameters as the hGRU (Fig. 3c, compare the red and green markers).

3.5 Residual networks

We reasoned that if the performance of feedforward models on the Pathfinder challenge is a function of model depth, then state-of-the-art networks for object recognition with many times the number of layers should easily solve the challenge. We tested this possibility by training ResNets with 18, 50, and 152 layers on the Pathfinder challenge. Each model was trained "from scratch" with the standard ResNet weight initialization, and given 4 epochs of training to learn the task due to their large number of parameters. However, even with this additional training time, only the deepest 152-layer ResNet was able to learn the task (Fig. 3a). The 152-layer ResNet was less efficient at learning the 14-path dataset than the hGRU (Fig. 3b), and achieved its performance on the with nearly $1000\times$ as many parameters (Fig. 3c).

3.6 Per-pixel prediction models

We considered the possibility that CNN architectures for per-pixel prediction tasks, such as contour detection and segmentation, might be better suited to the Pathfinder challenge than those designed for classification. We therefore tested three representative per-pixel prediction models: the fully-convolutional network (FCN), the skip-connection U-Net, and the unpooling SegNet. These models consisted of an encoder/decoder style architecture followed by the readout processing stage of the standard architecture described above. Encoders were the VGG16 [45], and each model was trained with random Xavier weight initializations. Like the ResNets, these models were given 4 epochs for training to accommodate their large number of parameters.

The fully-convolutional network (FCN) architecture is one of the first successful uses of CNNs for per-pixel prediction [3, 40, 46, 47]. Decoders in these models use "1×1" convolutions to combine upsampled activity maps from several layers of the encoder. We created an FCN model which applied this procedure to the last layer of each of the 5 VGG16-convolution blocks. These activity maps were upsampled by learnable kernels, which were initialized with weights for bilinear interpolation. In contrast to the feedforward models discussed above, the FCN successfully learned all conditions in

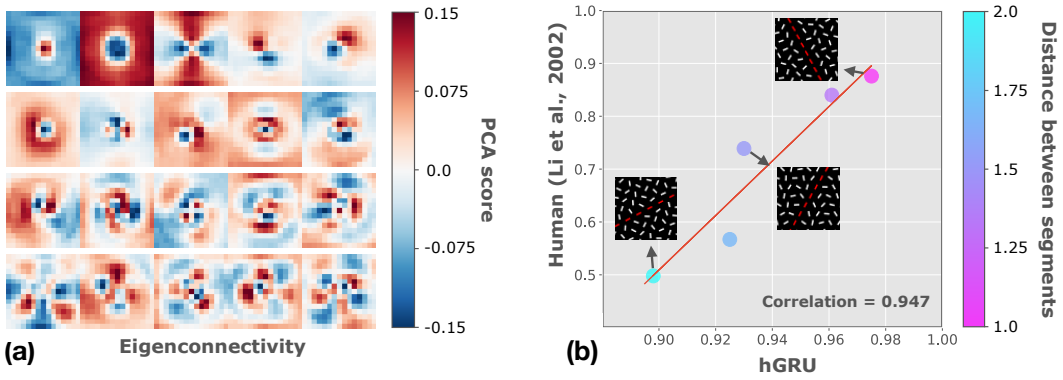

Figure 4: The hGRU learns horizontal connections that resemble cortical patterns of connectivity. (a) Processed kernels from the hGRU depict polar center-surround interactions and association fields. (b) An hGRU with these kernels that is fine-tuned on a classic contour detection task performs similarly to human observers. While the hGRU is overall more effective, the performance of both is strained as the distance between paddles increases.

the Pathfinder challenge (Fig. 3a, purple circle). It did so less efficiently than the hGRU, however, with a lower ALC score on the 14-length dataset as well as 200× as many free parameters (Fig. 3b).

Another approach to per-pixel prediction uses "skip connections" to connect specific layers of a model's encoder to its decoder. This approach was first described in [40] as a method for more effectively merging coarse-layer information into a model's decoder, and later extended to the U-Net [48]. We implemented a version of the U-Net architecture that had a VGG16 encoder and a decoder. The decoder consisted of 5 learnable, randomly initialized, upsampling layers, which had additive connections to the final convolutional layer in each of the encoder's VGG16 blocks. Using standard VGG16 nomenclature to define one of these connections, this meant that "conv 4_3" activity from the encoder was added to the second upsampled activity map in the decoder. The U-Net was on par with the hGRU and the FCN at solving the Pathfinder challenge. It was also nearly as efficient as the hGRU in doing so (Fig. 3b), but achieved its performance with over 350× as many parameters as the hGRU, (Fig. 3c).

Unpooling models eliminate the need for feature map upsampling by routing decoded activities to the locations of the winning max-pooling units derived from the encoder. Unpooling is also a leading approach for a variety of dense per-pixel prediction tasks, including segmentation, which is exemplified by SegNet [49]. We tested a SegNet on the Pathfinder challenge. This model has a decoder that mirrors its encoder, with unpooling operations replacing its pooling. The SegNet achieved high accuracy on each of the Pathfinder datasets, but was less efficient at learning them than the hGRU, with worse ALC scores across the challenge (Fig. 3b). The SegNet also featured the second-most parameters of any model tested, which was 400× more than the hGRU

4 Explaining biological horizontal connections with the hGRU

Statistical image analysis studies have suggested that cortical patterns of horizontal connections, commonly referred to as "association fields", may reflect the geometric regularities of oriented elements present in natural scenes [50]. We thus investigated whether training an hGRU on a contour detection task in natural scenes similarly lead to learned patterns of horizontal connections that resemble association fields.

Learning an association field from natural scenes We trained an hGRU to detect contours on the BSDS500 [1]. We preprocessed the learned kernels with a PCA-based procedure, and visualized the resulting "eigenconnectivity" patterns in Fig. 4a, which are sorted by their eigenvalues (80% cumulative variance cutoff; see SI for details on training and the processing used for visualization). Most prominent among these are (1) the antagonistic near-excitatory vs. far-inhibitory surround organization also found in the visual cortex [51], (2) the association field, with collinear excitation and orthogonal inhibition [8, 9], and (3) other higher-order surround computations [52]. Importantly, these patterns are qualitatively different from those learned on the synthetic Pathfinder dataset, which lack the strong center-surround or association field organization found in these filters (Suppl. Fig. 3).

Explaining human contour detection with association fields How well do these learned kernels explain human psychophysics data? We investigated this by recreating the synthetic dataset used in [18] to test human observers on a contour detection task. This task had participants detect a contour formed by co-linearly aligned paddles in an array of randomly oriented distractors. Multiple versions of the task were created by varying the distance between paddles in the contour (7 conditions), and varying the number of paddles making up the contour (4 conditions). We generated 1,000,000 unique 150×150 pixel images for every condition (28 total). In each case, 90% of images (900,000) were used for training and the remaining 10% (100,000) for testing. We fine-tuned our BSDS-trained hGRU separately on each of these training datasets and recorded its accuracy on the corresponding test datasets for comparison with human observers.

Unlike human observers [18], the hGRU was insensitive to the length of contours in the images. This tolerance to contour length is consistent with the model's behavior in Fig. 3, where the 8-timestep hGRU performed equally well on both 6- and 14-paddle Pathfinder datasets. While the hGRU also performed well when the distance between paddles increased, this manipulation nevertheless strained the network. We compared hGRU performance with human participants, whose responses were digitally extracted from [18] and averaged together for every experimental condition. Plotting the accuracy of the hGRU against the reported "detection score" of human observers revealed similar straining for both in response to increasing inter-paddle distance (Fig. 4b).

5 Discussion

Overall, the present study demonstrates that long-range spatial dependencies generally strain CNNs, with only very deep and state-of-the-art networks overcoming the visual variability introduced by long paths in the Pathfinder challenge. Although feedforward networks are generally effective at learning and detecting relatively rigid objects shown in well-defined poses, these models tend towards a brute-force solution when tasked with the recognition of less constrained structures, such as a path connecting two distant locations. This study adds to a body of work highlighting examples of routine visual task on which CNNs fall short of human performance [53–57].

We demonstrate a solution to the Pathfinder challenge inspired by neuroscience. The hGRU leverages computational principles of visual cortical circuits to learn complex spatial interactions between units. For the Pathfinder challenge, this translates into an ability to represent the elements forming an extended path while ignoring surrounding clutter. We find that the hGRU can reliably detect paths of any tested length or form using just a single layer. This contrasts sharply with the successful state-of-the-art feedforward alternatives, which used much deeper architectures with orders of magnitude more parameters to achieve similar success. The key mechanisms underlying the hGRU are well known in computational neuroscience [10, 12, 16]. However, to our knowledge, this is the first time that they are instantiated in an end-to-end trainable architecture that is effective for computer vision.

We also found that hGRU performance on the Pathfinder challenge is a function of the *amount of time* it was given for processing. This finding suggests that it gradually expands the excitatory influence of one end of a target curve to the other, while using inhibition to control the influence of distractors. The hGRU performance on these datasets captures the iterative nature of computations used by our visual system during Pathfinder tasks [58] – exhibiting a similar tradeoff between performance and processing-time as humans [7].

Visual cortex is replete with association fields that are thought to underlie perceptual grouping. Theoretical models suggest that patterns of horizontal connections reflect the statistics of natural scenes, and here too we find that horizontal kernels in the hGRU learned from natural scenes resemble cortical patterns of horizontal connectivity, including association fields and the paired near-excitatory and far-inhibitory surrounds that may underlie many contextual illusions [16, 51]. The horizontal connections learned by the hGRU trained on natural scenes reproduce another aspect of human behavior, in which saliency of a straight contour decreases as the distance between its paddles increases. This sheds light on a possible role that horizontal connections might play in saliency computation.

In summary, this work diagnoses a computational deficiency of feedforward networks, and introduces a biologically-inspired solution that can be easily incorporated into existing deep learning architectures. The weights and patterns of behavior learned by the hGRU appear consistent with those associated with the visual cortex, demonstrating its potential for establishing novel connections between machine learning, cognitive science, and neuroscience.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[2] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-Supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, February 2015.

[3] S. Xie and Z. Tu. Holistically-Nested edge detection. *Int. J. Comput. Vis.*, 125(1):3–18, December 2017.

[4] Y. Liu, M. M. Cheng, X. Hu, K. Wang, and X. Bai. Richer convolutional features for edge detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5872–5881, July 2017.

[5] K.-K. Maninis, J. Pont-Tuset, P. Arbelaez, and L. Van Gool. Convolutional oriented boundaries: From image segmentation to High-Level tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4): 819–833, April 2018.

[6] Y. Wang, X. Zhao, Y. Li, and K. Huang. Deep crisp boundaries: From boundaries to higher-level tasks. January 2018.

[7] R. Houtkamp and P. R. Roelfsema. Parallel and serial grouping of image elements in visual perception. *J. Exp. Psychol. Hum. Percept. Perform.*, 36(6):1443–1459, December 2010.

[8] D. D. Stettler, A. Das, J. Bennett, and C. D. Gilbert. Lateral connectivity and contextual interactions in macaque primary visual cortex. *Neuron*, 36(4):739–750, November 2002.

[9] K. S. Rockland and J. S. Lund. Intrinsic laminar lattice connections in primate visual cortex. *J. Comp. Neurol.*, 216(3):303–318, May 1983.

[10] S. Grossberg and E. Mingolla. Neural dynamics of perceptual grouping: textures, boundaries, and emergent segmentations. *Percept. Psychophys.*, 38(2):141–171, August 1985.

[11] D. J. Field, A. Hayes, and R. F. Hess. Contour integration by the human visual system: Evidence for a local "association field". *Vision Res.*, 33(2):173–193, 1993.

[12] G. W. Lesher and E. Mingolla. The role of edges and line-ends in illusory contour formation. *Vision Res.*, 33(16):2253–2270, November 1993.

[13] W. Li, V. Piëch, and C. D. Gilbert. Contour saliency in primary visual cortex. *Neuron*, 50(6): 951–962, June 2006.

[14] W. Li, V. Piëch, and C. D. Gilbert. Learning to link visual contours. *Neuron*, 57(3):442–451, February 2008.

[15] P. Series, J. Lorenceau, and Y. Frégnac. The "silent" surround of V1 receptive fields: theory and experiments. *Journal of physiology-Paris*, 97:453–474, 2003.

[16] D. A. Mely and T. Serre. Opponent surrounds explain diversity of contextual phenomena across visual modalities. August 2016.

[17] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN Encoder-Decoder for statistical machine translation. June 2014.

[18] W. Li and C. D. Gilbert. Global contour saliency and local colinear interactions. *J. Neurophysiol.*, 88(5):2846–2856, November 2002.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[20] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. November 2015.

[21] M. Siam, S. Valipour, M. Jagersand, and N. Ray. Convolutional gated recurrent networks for video segmentation. November 2016.

[22] Y. Shi, Y. Tian, Y. Wang, W. Zeng, and T. Huang. Learning Long-Term dependencies for action recognition with a Biologically-Inspired deep network. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 716–725, October 2017.

[23] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. May 2016.

[24] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc., 2009.

[25] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio. ReNet: A recurrent neural network based alternative to convolutional networks. May 2015.

[26] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2874–2883, 2016.

[27] L. Theis and M. Bethge. Generative image modeling using spatial LSTMs. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1927–1935. Curran Associates, Inc., 2015.

[28] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. January 2016.

[29] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *International Conference on Learning Representations*, 2016.

[30] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, April 2018.

[31] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537. IEEE, December 2015.

[32] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

[33] Q. Liao and T. Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. April 2016.

[34] J. Kim, J. K. Lee, and K. M. Lee. Deeply-Recursive convolutional network for image Super-Resolution. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645. IEEE, June 2016.

[35] C. J. Spoerer, P. McClure, and N. Kriegeskorte. Recurrent convolutional neural networks: A better model of biological object recognition. *Front. Psychol.*, 8:1551, September 2017.

[36] S. Shushruth, P. Mangapathy, J. M. Ichida, P. C. Bressloff, L. Schwabe, and A. Angelucci. Strong recurrent networks compute the orientation tuning of surround modulation in the primate primary visual cortex. *Journal of Neuroscience*, 32(1):308–321, 2012.

[37] D. B. Rubin, S. D. Van Hooser, and K. D. Miller. The stabilized supralinear network: a unifying circuit motif underlying multi-input integration in sensory cortex. *Neuron*, 85(2):402–417, January 2015.

[38] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, May 2004.

[39] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. March 2016.

[40] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[41] F. Yu and V. Koltun. Multi-Scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[42] T. Wang, M. Sun, and K. Hu. Dilated deep residual network for image denoising. August 2017.

[43] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka. Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. September 2017.

[44] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. November 2017.

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for Large-Scale image recognition. September 2014.

[46] G. Papandreou, I. Kokkinos, and P.-A. Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 390–399. IEEE, June 2015.

[47] K.-K. Maninis, J. Pont-Tuset, P. Arbelaez, and L. Van Gool. Convolutional oriented boundaries: From image segmentation to High-Level tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, May 2017.

[48] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing, 2015.

[49] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional Encoder-Decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2481–2495, December 2017.

[50] O. Ben-Shahar and S. Zucker. Geometrical computations explain projection patterns of long-range horizontal connections in visual cortex. *Neural Comput.*, 16(3):445–476, March 2004.

[51] S. Shushruth, L. Nurminen, M. Bijanzadeh, J. M. Ichida, S. Vanni, and A. Angelucci. Different orientation tuning of near- and far-surround suppression in macaque primary visual cortex mirrors their tuning in human perception. *J. Neurosci.*, 33(1):106–119, January 2013.

[52] H. Tanaka and I. Ohzawa. Surround suppression of V1 neurons mediates orientation-based representation of high-order visual features. *J. Neurophysiol.*, 101(3):1444–1462, March 2009.

[53] M. Ricci, J. Kim, and T. Serre. Not-So-CLEVR: Visual relations strain feedforward neural networks. February 2018.

[54] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. December 2013.

[55] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[56] A. Volokitin, G. Roig, and T. A. Poggio. Do deep neural networks suffer from crowding? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5628–5638. Curran Associates, Inc., 2017.

[57] K. Ellis, A. Solar-Lezama, and J. Tenenbaum. Unsupervised learning by program synthesis. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 973–981. Curran Associates, Inc., 2015.

[58] P. R. Roelfsema and R. Houtkamp. Incremental grouping of image elements in vision. *Atten. Percept. Psychophys.*, 73(8):2542–2572, November 2011.

[59] J. W. Peirce. PsychoPy—Psychophysics software in python. *J. Neurosci. Methods*, 162(1):8–13, May 2007.

[60] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 2010. PMLR.

[61] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. December 2014.

[62] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[63] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. February 2015.

Supplementary Material

6 The Pathfinder challenge

Overview The main goal of the Pathfinder challenge is to assess the ability of a computer vision system to determine whether two distant locations in an image, marked by while circles, are connected by a path. The stimulus dataset consists of binary images, each containing two white circles, two white 'target' paths, and multiple distractor paths, placed on a black background. All images used in our experiments are rendered on a 150×150 canvas. The task is inspired by [58].

Each image is generated by first sampling two target paths and then rendering multiple distractor paths. Each path consists of k co-circularly arranged identically shaped "paddles" of length l and thickness d (Fig. 6b). A paddle is a white oriented bar, characterized by the position of its center and its orientation, θ. Each path is generated inductively, namely, by first sampling the position of its "seed paddle" which serves as the end of the path and then iteratively adding new "trailing paddles" next to the seed paddle or the last trailing paddle. The high-level overview of the image generation algorithm is depicted in Fig. 6 and the list of image parameters in Table 6.

Positioning target paths The first stage of generating target paths involves randomly sampling the position of an invisible circle of radius r (Fig. 6a, step 1) within an empty image. Then, a randomly oriented line pivoted at the center of the circle makes two intersections with the circle. These intersections serve as the positions of two target paths' seed paddles (Fig. 6a, step 2). This constraint ensures that target paths are always located within a sufficient proximity to each other. Without this constraint, target paths can be separated by an arbitrary distance, which may allow a model to make classification decision solely based on the distance between the white markers, for the markers will tend to be located much farther apart in negative examples. Then, two randomly oriented paddles are rendered at each of the intersections (Fig. 6a, step 3), serving as seed paddles of the target paths.

Growing a path Trailing paddles are sequentially added to the end of each target path (Fig. 6a, step 4 and 5). Each trailing paddle is added by first sampling its orientation, θ_i, according to the probability distribution defined by the angle formed between the new and the previous trailing paddle, $\Delta\theta = \theta_i - \theta_{i-1}$:

$$P(\theta_i) = \frac{1}{Z}\max(\cos\left(c\Delta\theta\right), 0) \tag{12}$$

$$\text{where } Z = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos(\theta)d\theta = 2 \tag{13}$$

Note that the continuity parameter c determines the overall rigidity of a path by constraining the possible range of orientations of each new trailing paddle. With the sampled orientation, the new trailing paddle's position is determined such that the line extending from the trailing paddle intersects with the line extending from the new paddle m pixels away from the end of the two paddles while also forming an angle $\Delta\theta$ (Fig. 6b). Note that the parameter m determines the margin between adjacent paddles in a path. Because we do not allow paddles to cross or touch, we add new paddles to the two target paths in alternation to ensure that the shapes of one path does not restrict the shape of the other. The total number of target path paddles, or simply length of target paths, is denoted by the parameter k.

Additional distractor paths consisting of $\frac{k}{3}$ paddles are added to the image. Their generation process is identical to that of target paths, except that a seed paddle's location is independently sampled. The total number of distractor paths is chosen so that the total number of paddles in an image is 150. The usage of two target paths as well as distractor paths prevents any 'short-cut' solution to this problem, such as detecting a loose end of a path or a lone white circle as a local diagnostic cue for classification. Additionally, this design allows us to ensure that positive and negative images are indistinguishable

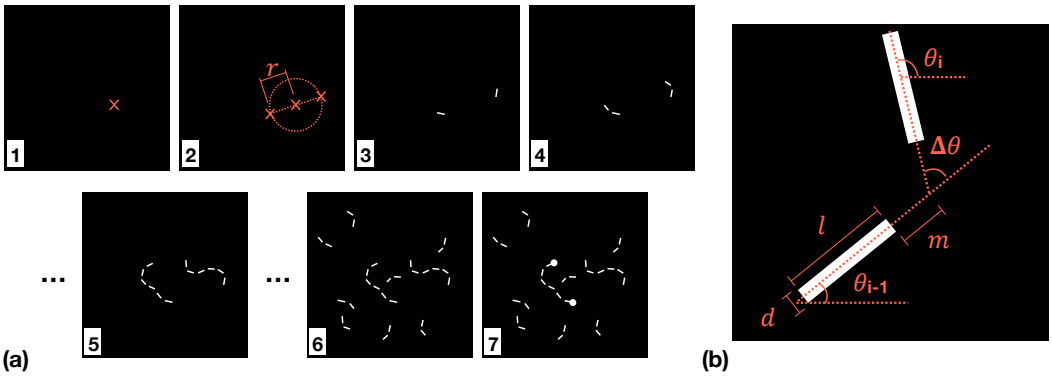

Figure S1: Image generator algorithm. (a) Depiction of image generation shown in seven steps. (b) Depiction of the geometric constraints for trailing paddle placement during path generation.

Table S1: Image parameters in the Pathfinder challenge.

Notation	Definition
r	Radius of a circle
k	Target path length, or the number of paddles in a target path
l	Paddle length, in pixels
d	Paddle thickness, in pixels
m	Inter-paddle margin, in pixels
c	Continuity

in terms of the way paddles are arranged. This forces a model to perform classification solely based on the connectedness of the white circles.

As mentioned, the generator does not allow paddles to be too close to each other or make any contact. If a newly sampled paddle is making contact with or not separated from other paddles by at least m pixels, the generator rejects it and re-samples a paddle. In practice, this is done by applying circular dilation on the paddles using a kernel of radius m and checking if the dilated images of paddles has any overlap.

Shape variability Because adding an additional paddle to a path multiplicatively increases the number of possible shapes of the path, path shape variability is exponential to the total number of paddles in a path, k. Similarly, the continuity parameter c controls path shape variability in an exponential manner because it scales the range of possible angles formed between every pair of adjacent paddles in a path. In our experiment, we vary path length k to examine how model performance changes as the shape variability of a path in a dataset increases.

All images in the Pathfinder challenge are generated using Python and OpenCV.

7 The Synthetic Contours Dataset

Overview In this section, we shall explain the motivation and construction procedure for the Synthetic Contours Dataset. We created the Synthetic Contours Dataset (SCD) to compare an hGRU pre-trained on the BSDS500 [1] for contour detection against human behavioral data as described in [18].

Dataset generation We generated the SCD by following closely, and extending the dataset construction procedure described in [18]. SCD enables us to parametrically control the position, orientation, and relative spacing of generated contours, resulting in 28 million images with practically infinite variability along these directions. All images in the SCD have a resolution of 256×256 pixels, spanning a radius of 4 visual degrees. Each contour image is rendered by placing "paddles" of length 0.1 visual degrees within uniformly spaced cells on a master grid. This master grid covers the entire

image with a height and a width of 8 visual degrees. Each cell in the grid spans a height and width of 0.25 visual degrees. A total of 32 paddles are placed along every row/column of the master grid.

A contour of length l is generated in the master grid by filling l neighboring grid cells on any grid-diagonal, and each paddle connects the diagonally opposite points in their respective cells. This aligns the paddles in a collinear manner to form a contour of length l. In order to ensure that the majority of the contour stays within the image, its center is positioned within a maximum distance from the center of the image. This distance from the image center, known as eccentricity, is denoted as e. Once the contour is generated, all remaining unoccupied cells are filled with paddles each with an orientation $\theta \in [0, 2\pi]$ sampled from a uniform distribution.

Parametric dataset variability Contours in SCD are formed by regulating the dataset variability in 2 different directions: (1) contour length and (2) relative spacing. We denote the number of paddles present in a contour as its length. Each image in the SCD contains a contour with a length of either 5, 9, 14 or 17 paddles.

The distance between neighboring paddles that form a contour is denoted as the relative spacing. While we vary the relative spacing, the global spacing between distractor paddles is maintained constant. Following [18], we apply a shear operation to the generated image by a shear factor *SF*, which controls how close or far the neighboring contour paddles are positioned. We generated images with 7 different relative-spacing conditions by varying SF between -0.6 and +0.6. A comparison between human performance and model performance on the task of contour detection can be found in the main paper (Fig.4b).

Example figures with different combinations of contour length and relative spacing are demonstrated in Fig. S2. All the images in SCD were generated using the Python Psychophysics toolbox, Psychopy [59].

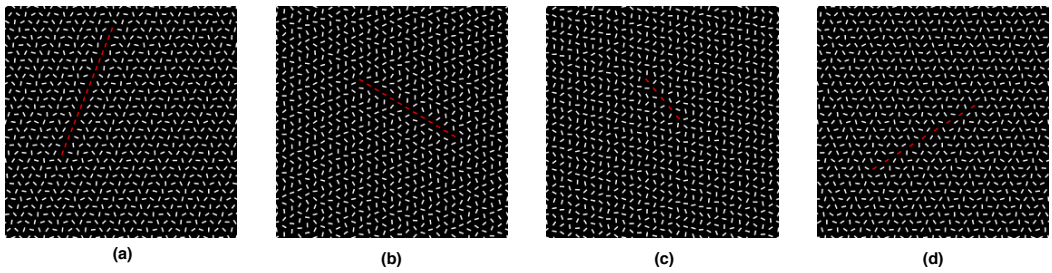

Figure S2: Examples of images from the SCD. Unlike in the dataset, contours in the figure are highlighted with a different color for the reader's convenience. (a) Contour with *SF*=-0.6, *l*=17, closest relative spacing. (b) Contour with *SF*=-0.4, *l*=14. (c) Contour with *SF*=0.2, *l*=9. (d) Contour with *SF*=0.6, *l*=9, farthest relative spacing.

8 Model architecture details

All of the recurrent models and feedforward models (shown in shades of green in Fig. 3a c in main paper) tested in our study share the 'input stage' as the preprocessing module and the 'readout stage' as the classification module, which allows us to more directly compare the effectiveness of different architectures (called 'feature extraction' stage) at solving the Pathfinder challenge. Unless otherwise specified, all model weights were Xavier initialized [60]. Standard cross entropy was used to compute loss. Models were trained with gradient descent using the Adaptive Moment Estimation (Adam) optimizer [61] with base learning rate of 10^{-3} on batches of 32 images. Our experiments are conducted using TensorFlow [62].

Input stage The input stage consists of a convolutional layer with 25 kernels of size 7×7. The kernels are initialized as Gabor filters at 12 orientations and 2 phases, plus a radially symmetric difference-of-Gaussian filter. Filter activity undergoes a point-wise squaring non-linearity before being passed to the model-specific feature extraction stage. Note that none of the off-the-shelf models (residual networks and per-pixel prediction models) use this input stage.

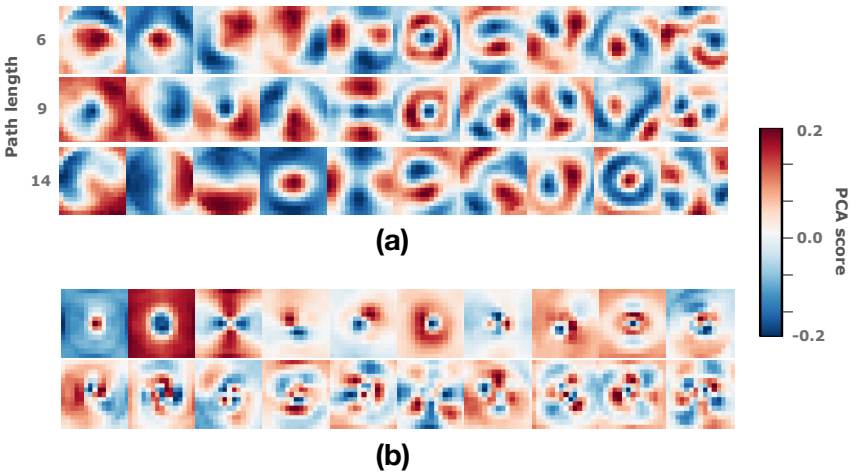

Figure S3: hGRU eigenconnectivity on natural scenes is qualtitatively different than synthetic ones. (a) Eigenconnectivity of hGRU models trained on 6-, 9-, and 14-length Pathfinder datasets. (b) Eigenconnectivity of an hGRU on the BSDS500 contour detection dataset. The exhibited patterns are similar to those observed in visual cortex.

Readout stage The readout stage takes the output from the final layer of the feature extraction stage and computes a two-dimensional likelihood vector for deriving a decision on an image from the Pathfinder challenge. The readout stage contained three parts: (1) a 1x1 convolutional filter that transforms the multi-channel output map from the feature extraction stage to a two-channel map, each corresponding to the positive/negative class. (2) A batch-normalized [63] global max pool, whose output represents in each channel the maximum activation value across space. (3) A linear classifier which maps the two-dimensional feature vector into a class decision. The configuration of this readout stage allows us to compare the accuracy between models designed for per-image prediction with those designed for per-pixel prediction. Note that residual networks do not use a readout stage as they are proposed as standalone image classification architectures.

Recurrent models GRU was originally designed to handle time-varying input data. As a result, input $\mathbf{X}$ is used at every timestep to compute its gates and hidden state (see main text). In practice, we found identical model performance when $\mathbf{X}$ was included or excluded from gate computations.

9 Training on natural images and comparisons to human data

Theoretical models of visual cortex suggest that patterns of horizontal connections reflect the statistics of natural scenes [50]. Here too we find that horizontal kernels in the hGRU learned from natural scenes resemble cortical patterns of horizontal connectivity, including association fields and the paired near-excitatory and far-inhibitory surrounds (Fig. 9b).

We trained an hGRU to detect contours on the BSDS500 [1]. The model was the 8-timestep hGRU with an input processing stage that was the first two blocks of PASCAL weight initialized convolutional layers from the VGG16. The model was trained for 1000 epochs on the 200 training images in the BSDS500, using data augmentations (random crops to 300×300 pixels, random rotations of +/- 30°, and random up/down/left/right flips), a per-pixel cross-entropy loss, and the same learning rate and optimizer as the models discussed above.

Patterns of connectivity in the hGRU kernels were visualized with a two-stage procedure. First, each was rotated into a common frame of reference by first fitting a Gaussian and then rotating the kernel to offset any angle detected in the fitting procedure. Second, we mean centered before applying PCA to the learned kernels and visualized the resulting "eigenconnectivity" patterns in Fig. 4a (main text) and Fig. 9. Eigenconnectivity patterns are sorted by their eigenvalues (80% cumulative variance cutoff).

Note that the eigenconnectivity patterns extracted from the kernels trained on natural images (Fig. 9b) differ significantly from the kernels trained on the Pathfinder challenge (Fig. 9a). For example, one

of the most marked differences between the two eigenconnectivity patterns is the presence/absence of center-surround structure (note the first two eigenconnectivity matrices in Fig. 9b.). We believe that such difference partially results from the difference in the kinds of natural image statistics that are useful for the respective tasks – contour detection and path integration –, for path integration task in the Pathfinder challenge can be solved by relying nearly exclusively on the co-linearity of neighboring paddles whereas contour detection in natural images might oftentimes require richer, sometimes non-directional, measures of local change.