

# Goal-Driven Recurrent Neural Network Models of the Ventral Visual Stream

Aran Nayebi<sup>1\*</sup>, Javier Sagastuy-Brena<sup>1</sup>, Daniel M. Bear<sup>1</sup>, Kohitij Kar<sup>2</sup>, Jonas Kubilius<sup>2,3</sup>, Surya Ganguli<sup>1</sup>, David Sussillo<sup>1</sup>, James J. DiCarlo<sup>2</sup>, Daniel L. K. Yamins<sup>1</sup>

<sup>1</sup> Stanford University, Stanford, USA

<sup>2</sup> Massachusetts Institute of Technology, Cambridge, USA

<sup>3</sup> KU Leuven, Leuven, Belgium

\*To whom correspondence should be addressed; E-mail: [anayebi@stanford.edu](mailto:anayebi@stanford.edu)

1        **The ventral visual stream (VVS) is a hierarchically connected series of cortical areas**        1  
2        **known to underlie core object recognition behaviors, enabling humans and non-human**        2  
3        **primates to effortlessly recognize objects across a multitude of viewing conditions. While**        3  
4        **recent feedforward convolutional neural networks (CNNs) provide quantitatively accurate**        4  
5        **predictions of temporally-averaged neural responses throughout the ventral pathway, they**        5  
6        **lack two ubiquitous neuroanatomical features: local recurrence within cortical areas and**        6  
7        **long-range feedback from downstream areas to upstream areas. As a result, such models**        7  
8        **are unable to account for the temporally-varying dynamical patterns thought to arise from**        8  
9        **recurrent visual circuits, nor can they provide insight into the behavioral goals that these**        9  
10       **recurrent circuits might help support. In this work, we augment CNNs with local recur-**        10  
11       **rence and long-range feedback, developing convolutional RNN (ConvRNN) network mod-**        11  
12       **els that more correctly mimic the gross neuroanatomy of the ventral pathway. Moreover,**        12

13 **when the form of the recurrent circuit is chosen properly, ConvRNNs with comparatively** 13  
14 **small numbers of layers can achieve high performance on a core recognition task, com-** 14  
15 **parable to that of much deeper feedforward networks. We then compared these models** 15  
16 **to temporally fine-grained neural and behavioral recordings from primates to thousands** 16  
17 **of images. We found that ConvRNNs better matched these data than alternative models,** 17  
18 **including the deepest feedforward networks, on two metrics: 1) neural dynamics in V4** 18  
19 **and inferotemporal (IT) cortex at late timepoints after stimulus onset, and 2) the varying** 19  
20 **times at which object identity can be decoded from IT, including more challenging im-** 20  
21 **ages that take longer to decode. Moreover, these results differentiate within the class of** 21  
22 **ConvRNNs, suggesting that there are strong functional constraints on the recurrent con-** 22  
23 **nectivity needed to match these phenomena. Finally, we find that recurrent circuits that** 23  
24 **attain high task performance while having a smaller network size as measured by number** 24  
25 **of units, rather than another metric such as the number of parameters, are overall most** 25  
26 **consistent with these data. Taken together, our results evince the role of recurrence and** 26  
27 **feedback in the ventral pathway to reliably perform core object recognition while subject** 27  
28 **to a strong total network size constraint.** 28

## 29 **1 Introduction** 29

30 The visual system of the brain must discover meaningful patterns in a complex physical world<sup>1</sup>. 30  
31 Within 200ms, primates can quickly identify objects despite changes in position, pose, contrast, 31  
32 background, foreground, and many other factors from one occasion to the next: a behavior 32  
33 known as “core object recognition”<sup>2,3</sup>. It is known that the ventral visual stream (VVS) under- 33  
34 lies this ability by transforming the retinal image of an object into a new internal representation, 34  
35 in which high-level properties, such as object identity and category, are more explicit<sup>3</sup>. 35

36 Task-optimized, deep convolutional neural networks (CNNs) are currently the most quanti- 36

37 tatively accurate models of encoding in primate visual cortex<sup>4,5,6</sup>. This observed correspondence 37  
38 is due to their cortically-inspired architecture, which consists of a cascade of spatially-tiled lin- 38  
39 ear and nonlinear operations; and their being optimized to perform the same behaviors that 39  
40 animals must perform to survive, such as object recognition<sup>7</sup>. CNNs trained to recognize ob- 40  
41 jects in the ImageNet dataset predict the *time-averaged* neural responses of cortical neurons 41  
42 better than any other model class. Model units from early, intermediate, and higher convolu- 42  
43 tional layers, respectively, provide the best-known linear predictions of time-averaged visual 43  
44 responses in neurons of early (area V1<sup>5,8</sup>), intermediate (area V4<sup>4</sup>), and higher visual cortical 44  
45 areas (inferotemporal cortex, IT<sup>4,5</sup>). 45

46 While these results are promising, it is not obvious how to extend the architecture and task- 46  
47 optimization of CNNs to the case where responses change over time. Non-trivial dynamics 47  
48 result from biological features not present in purely feedforward CNNs, including synapses 48  
49 that facilitate or depress, dense local recurrent connections within each cortical region, and 49  
50 long-range connections between different regions, such as feedback from higher to lower visual 50  
51 cortex<sup>9</sup>. Furthermore, the behavioral roles of recurrence and dynamics in the visual system 51  
52 are not well understood. Several conjectures are that recurrence “fills in” missing data,<sup>10,11,12,13</sup> 52  
53 such as object parts occluded by other objects; that it “sharpens” representations by top-down 53  
54 attentional feature refinement, allowing for easier decoding of certain stimulus properties or 54  
55 performance of certain tasks<sup>9,14,15,16,17</sup>; that it allows the brain to “predict” future stimuli (such 55  
56 as the frames of a movie)<sup>18,19,20</sup>; or that recurrence “extends” a feedforward computation, re- 56  
57 flecting the fact that an unrolled recurrent network is equivalent to a deeper feedforward net- 57  
58 work that conserves on neurons (and learnable parameters) by repeating transformations several 58  
59 times<sup>21,22,23,12</sup>. Formal computational models are needed to test these hypotheses: if optimizing 59  
60 a model for a certain task leads to accurate predictions of neural dynamics, then that task may 60  
61 be a primary reason those dynamics occur in the brain. 61

62 We therefore broaden the method of goal-driven modeling from solving tasks with feedfor- 62  
63 ward CNNs<sup>7</sup> or RNNs<sup>24</sup> to explain dynamics in the primate visual system, resulting in con- 63  
64 volutional recurrent neural networks (ConvRNNs). We show that with the appropriate choice 64  
65 of layer-local recurrence and feedback connections, ConvRNNs can match the performance of 65  
66 much deeper feedforward CNNs on ImageNet but with far fewer parameters and a more anatom- 66  
67 ically consistent number of layers. Furthermore, we found that such task-optimized ConvRNNs 67  
68 better match the VVS than feedforward CNNs by two metrics: 1) they are able to match the 68  
69 fine-timescale trajectories of neural responses in the visual pathway across the *entirety* of stim- 69  
70 ulus presentation with a fixed linear mapping, and 2) they provide a better match to primate 70  
71 behavior in the form of object solution times. Specifically, we observe that ConvRNNs that 71  
72 attain high task performance but low network size, as measured by number of units, are most 72  
73 consistent with both of these metrics. These results indicate that very deep feedforward models 73  
74 are overall a *less consistent* match to primate VVS than shallower feedforward networks with 74  
75 added recurrence. This in turn implies that the role of recurrence in core object recognition 75  
76 is consistent with the hypothesis of extending a shallower feedforward network across time in 76  
77 order to perform a categorization task while obeying a physical size constraint. 77

## 78 **2 Results** 78

### 79 **2.1 An evolutionary architecture search yields specific layer-local recur- 79** 80 **rent circuits and long-range feedbacks that improve task performance.** 80

81 We first tested whether augmenting CNNs with standard RNN circuits from the machine learn- 81  
82 ing community, SimpleRNNs and LSTMs, could improve performance on ImageNet object 82  
83 recognition (Figure 2a). We found that these recurrent circuits<sup>a</sup> added a small amount of ac- 83  
84 curacy when introduced into the convolutional layers of a 6-layer feedforward backbone (“FF” 84

---

<sup>a</sup>Adapting other recurrent cell structures to ConvRNNs from the literature, including the UGRNN and Inter-  
sectionRNN<sup>25</sup>, had similar effects.



85 in Figure 2b) based off of the AlexNet<sup>26</sup> architecture, which we will refer to as a “BaseNet” 85  
86 (see Section A.3 for architecture details). However, there were two problems with these re- 86  
87 sultant recurrent architectures: first, these ConvRNNs did not perform substantially better than 87  
88 parameter-matched, minimally unrolled controls – defined as the minimum number of timesteps 88  
89 after the initial feedforward pass whereby all recurrence connections were engaged at least 89  
90 once. This control comparison suggested that the observed performance gain was due to an in- 90  
91 crease in the number of unique parameters added by the implanted ConvRNN cells rather than 91  
92 temporally-extended recurrent computation. Second, making the feedforward model wider or 92  
93 deeper yielded an even larger performance gain than adding these standard RNN cells, but with 93  
94 fewer parameters. This suggested that standard RNN circuits, although well-suited for a range 94  
95 of temporal tasks, are less well-suited for inclusion within deep CNNs to solve challenging 95  
96 object recognition tasks. 96

97 We speculated that this was because standard circuits lack a combination of two key proper- 97  
98 ties, each of which on their own have been successful either purely for RNNs or for feedforward 98  
99 CNNs: (1) **Gating**, in which the value of a hidden state determines how much of the bottom-up 99  
100 input is passed through, retained, or discarded at the next time step; and (2) **Bypassing**, where 100  
101 a zero-initialized hidden state allows feedforward input to pass on to the next layer unaltered, 101  
102 as in the identity shortcuts of ResNet-class architectures (Figure 2a; top left). For example, 102  
103 LSTMs employ gating, but no bypassing, as their inputs must pass through several nonlinear- 103  
104 ities to reach their output; whereas SimpleRNNs do bypass a zero-initialized hidden state, but 104  
105 do not gate their input (Figure 2a). 105

106 We thus implemented recurrent circuits with both features to determine whether they func- 106  
107 tion better than standard cells within CNNs. One example of such a structure is the “Reciprocal 107  
108 Gated Cell” (RGC)<sup>27</sup>, which bypasses its zero-initialized hidden state and incorporates LSTM- 108  
109 style gating (Figure 2a, bottom right; see Section A.3.7 for the cell equations). Adding this 109

110 cell to the 6-layer BaseNet improved performance substantially relative to both the feedforward 110  
111 baseline and minimally unrolled, parameter-matched control version of this model. Moreover, 111  
112 the RGC used substantially fewer parameters than the standard cells to achieve greater accuracy 112  
113 (Figure 2b). 113

114 However, it has been shown that different RNN structures can succeed or fail to perform a 114  
115 given task because of differences in trainability rather than differences in capacity<sup>25</sup>. Therefore, 115  
116 we designed an evolutionary search to jointly optimize over both discrete choices of recurrent 116  
117 connectivity patterns as well as continuous choices of learning hyperparameters and weight 117  
118 initializations (search details in Section A.4). While a large-scale search over thousands of con- 118  
119 volutional LSTM architectures did yield a better purely gated LSTM-based ConvRNN (“LSTM 119  
120 Opt”), it did not eclipse the performance of the smaller RGC ConvRNN. In fact, applying the 120  
121 same hyperparameter optimization procedure to the RGC ConvRNNs equally increased that ar- 121  
122 chitecture class’s performance and further reduced its parameter count (Figure 2b, “RGC Opt”). 122

123 If the primate visual system uses recurrence in lieu of greater network depth to perform 123  
124 object recognition, then a shallower recurrent model with a suitable form of recurrence should 124  
125 achieve recognition accuracy equal to a deeper feedforward model, albeit with temporally-fixed 125  
126 parameters<sup>21</sup>. We therefore tested whether our search (depicted in Figure 2c) had identified such 126  
127 well-adapted recurrent architectures by fully training a representative ConvRNN, the model 127  
128 with the median five-epoch performance after 7000 samples. This median model (“RGC Me- 128  
129 dian”) reached a final ImageNet top-1 validation accuracy nearly equal to a ResNet-34 model 129  
130 with nearly twice as many layers, even though the ConvRNN used only  $\sim 75\%$  as many pa- 130  
131 rameters. The fully unrolled model from the random phase of the search (“RGC Random”) did 131  
132 not perform substantially better than the BaseNet, though the minimally unrolled control did 132  
133 (Figure 2d). This observation suggests that our evolutionary search strategy yielded effective 133  
134 recurrent architectures beyond the initial random phase of the search. 134

135 We also considered a control model (“Time Decay”) that produces temporal dynamics by 135  
136 learning time constants on the activations independently at each layer, rather than by learning 136  
137 connectivity between units. In this ConvRNN, unit activations have exponential rather than 137  
138 immediate falloff once feedforward drive ceases. These dynamics could arise, for instance, 138  
139 from single-neuron biophysics (e.g. synaptic depression) rather than interneuronal connections. 139  
140 However, this model did not perform any better than the feedforward BaseNet, implying that 140  
141 ConvRNN performance is not a trivial result of outputting a dynamic time course of responses. 141  
142 We further implanted other more sophisticated forms of ConvRNN cells into the BaseNet, and 142  
143 while this improved performance over the Time Decay model, it did not outperform the RGC 143  
144 Median ConvRNN despite having many more parameters (Figure 2d). Together, these results 144  
145 demonstrate that the RGC Median ConvRNN uses recurrent computations to subserve object 145  
146 recognition behavior and that particular motifs in its recurrent architecture (Figure S1), found 146  
147 through an evolutionary search, are required for its improved accuracy. Thus, given suitable 147  
148 local recurrent circuits and patterns of long-range feedback connectivity, a physically more 148  
149 compact, temporally-extended ConvRNN can do the same challenging object recognition task 149  
150 as a deeper feedforward CNN. 150

## 151 **2.2 ConvRNNs provide an improved explanation of neural dynamics.** 151

152 ConvRNNs naturally produce a dynamic time series of outputs given an unchanging input 152  
153 stream, unlike feedforward networks. While these recurrent dynamics could be used for tasks 153  
154 involving time, here we optimized the ConvRNNs to perform the “static” task of object classi- 154  
155 fication on ImageNet. It is possible that the primate visual system is optimized for such a task, 155  
156 because even static images produce reliably dynamic neural response trajectories at temporal 156  
157 resolutions of tens of milliseconds<sup>20</sup>. The object content of some images becomes decodable 157  
158 from the neural population significantly later than the content of other images, even though an- 158

159 imals recognize both object sets equally well. Interestingly, late-decoding images are not well 159  
160 characterized by feedforward CNNs, raising the possibility that they are encoded in animals 160  
161 through recurrent computations<sup>17</sup>. If this were the case, we reason then that recurrent networks 161  
162 trained to perform a difficult, but static object recognition task might explain neural dynamics 162  
163 in the primate visual system, just as feedforward models explain time-averaged responses<sup>4,5</sup>. 163

164 Prior studies<sup>28</sup> have *directly* fit recurrent parameters to neural data, as opposed to optimizing 164  
165 them on a task. While it is natural to try to fit recurrent parameters to the temporally-varying 165  
166 neural responses directly, we found that this approach suffers from a fundamental overfitting 166  
167 issue to the particular image statistics of the neural data collected. Specifically, we directly 167  
168 fit these recurrent parameters (implanted into the task-optimized feedforward BaseNet) to the 168  
169 dynamic firing rates of primate neurons recorded during encoding of visual stimuli. However, 169  
170 while these non-task optimized dynamics generalized to held-out images and neurons (Fig- 170  
171 ure S2a,b), they had no longer retained performance to the original object recognition task 171  
172 (Figure S2c). Therefore, to avoid this issue, we instead asked whether *fully* task-optimized 172  
173 ConvRNN models (including the ones introduced in Section 2.1) could predict these dynamic 173  
174 firing rates from multi-electrode array recordings from the ventral visual pathway of rhesus 174  
175 macaques<sup>29</sup>. 175

176 We began with the feedforward BaseNet and added a variety of ConvRNN cells, including 176  
177 the RGC Median ConvRNN and its counterpart generated at the random phase of the evolu- 177  
178 tionary search (“RGC Random”). All of the ConvRNNs were presented with the same images 178  
179 shown to the primates and collected the time series of features from each model layer. To decide 179  
180 which layer should be used to predict which neural responses, we fit linear models from each 180  
181 feedforward layer’s features to the neural population and measured where explained variance on 181  
182 held-out images peaked (see Section A.6 for more details). Units recorded from distinct arrays 182  
183 – placed in the successive V4, posterior IT (pIT), and central/anterior IT (cIT/aIT) cortical areas 183

184 of the macaque – were fit best by the successive layers of the feedforward model, respectively. 184  
185 Finally, we measured how well ConvRNN features from these layers predicted the dynamics 185  
186 of each unit. In contrast with feedforward models fit to temporally-averaged neural responses, 186  
187 the linear mapping in the temporal setting must be *fixed* at all timepoints. The reason for this 187  
188 choice is that the linear mapping yields “artificial units” whose activity can change over time, 188  
189 but the identity of these units should not change over the course of 260ms, as would be the case 189  
190 if a separate linear mapping was fit at each 10ms timebin. This choice of a temporally-fixed 190  
191 linear mapping therefore maintains the physical relationship between real neurons and model 191  
192 neurons. 192

193 As can be seen from Figure 3, with the exception of the RGC Random ConvRNN, the Con- 193  
194 vRNN feature dynamics fit the neural response trajectories as well as the feedforward baseline 194  
195 features on early phase responses (Wilcoxon test  $p$ -values in Table 1) and better than the feed- 195  
196 forward baseline features for late phase responses (Wilcoxon test with Bonferroni correction 196  
197  $p < 0.001$ ), across V4, pIT, and cIT/aIT on held-out images. This observation is due to the 197  
198 fact that any feedforward model has the same square wave dynamics as its 100ms visual input, 198  
199 so it cannot predict neural responses after image offset plus a fixed delay, corresponding to the 199  
200 number of layers (Figure S3, purple lines). In contrast, the activations of ConvRNN units have 200  
201 persistent dynamics, yielding predictions of the *entire* neural response trajectories. 201

202 Crucially, these predictions result from the task-optimized nonlinear dynamics from Ima- 202  
203 geNet, as both models are fit to neural data with the same form of temporally-fixed linear model 203  
204 with the *same* number of parameters. Since the initial phase of neural dynamics was well-fit by 204  
205 feedforward models, we asked whether the later phase could be fit by a much simpler model 205  
206 than any of the ConvRNNs we considered, namely the Time Decay ConvRNN with ImageNet- 206  
207 trained time constants at convolutional layers. If the Time Decay ConvRNN were to explain 207  
208 neural data as well as the other ConvRNNs, it would imply that interneuronal recurrent con- 208

nections are not needed to account for the observed dynamics; however, this model did not fit the late phase dynamics of intermediate areas (V4 and pIT) as well as the other ConvRNNs<sup>b</sup>. Thus, the more complex recurrence found in ConvRNNs is generally needed both to improve object recognition performance and to account for neural dynamics in the ventral stream, even when animals are only required to fixate on visual stimuli. In fact, not all forms of complex recurrence are equally predictive of temporal dynamics. We found among these that the RGC Median, UGRNN, and GRU ConvRNNs attained the highest median neural predictivity for each visual area in both early and late phases, but in particular significantly outperformed the SimpleRNN ConvRNN at the late phase dynamics of these areas<sup>c</sup>. We will explore this observation further in Section 2.3.

A natural follow-up question to ask is whether recurrent processing explains any more of the variance at any *individual* timebins than feedforward models, especially in light of the observation that there is a drop in explained variance for feedforward models from early to late timebins<sup>17</sup>. It is well-known that recurrent neural networks can be viewed as very deep feedforward networks with weight sharing across layers that would otherwise be recurrently connected<sup>21</sup>. In Figure 3, we present the feedforward BaseNet with a constant stream of inputs in order for it to have a consistent output throughout time, despite the ConvRNNs and the primates only being provided with a 100ms stimulus presentation. We find that while the BaseNet in this setting can slightly outperform the best ConvRNNs at the late phase dynamics (Wilcoxon test with Bonferroni correction  $p < 0.001$ ), it underperforms relative to itself and the best ConvRNNs at the early phase dynamics (Wilcoxon test with Bonferroni correction  $p < 0.001$ ), providing not as consistent predictions under a temporally-fixed mapping. Thus, to address this question, we compare feedforward models of varying depths (with a constant input stream) to

---

<sup>b</sup>Wilcoxon test with Bonferroni correction  $p < 0.001$  for each ConvRNN vs. Time Decay, except for the SimpleRNN  $p \approx 0.46$  for pIT.

<sup>c</sup>Wilcoxon test with Bonferroni correction between each of these ConvRNNs vs. the SimpleRNN on late phase dynamics,  $p < 0.001$  per visual area.

232 ConvRNNs across the entire temporal trajectory under a *varying* linear mapping at each time- 232  
233 bin, in contrast to the above. Specifically, as can be seen in Figure S4a, we observe a drop 233  
234 in explained variance from early (130-140ms) to late (200-210ms) timebins for the shallower 234  
235 BaseNet and ResNet-18 models, across multiple neural datasets. Models with increased feed- 235  
236 forward depth (such as ResNet-101 or ResNet-152), along with our performance-optimized 236  
237 RGC Median ConvRNN, exhibit a similar drop in median population explained variance as the 237  
238 shallower feedforward models. The benefit of model depth with respect to increased explained 238  
239 variance of late IT responses might be only noticeable while comparing very shallow models 239  
240 ( $< 7$  nonlinear transforms) to much deeper ( $> 15$  nonlinear transforms) models<sup>17</sup>. Our results 240  
241 suggest that the amount of variance explained in the late IT responses is not a monotonically 241  
242 increasing function of model depth. 242

243 As a result, an alternative hypothesis is that the drop in explained variance from early to 243  
244 late timebins could instead be attributed to task-orthogonal dynamics specific to an individ- 244  
245 ual primate as opposed to iterated nonlinear transforms, resulting in variability unable to be 245  
246 captured by any task-optimized model (feedforward or recurrent). To explore this possibility, 246  
247 we examined whether the model's neural predictivity at these early and late timebins was rel- 247  
248 atively similar in ratio to that of one primate's IT neurons mapped to that of another primate 248  
249 (see Section A.7 for more details). As shown in Figure S4b, across various hyperparameters 249  
250 of the linear mapping, we observe a ratio close to one between the neural predictivity (of the 250  
251 target primate neurons) of the feedforward BaseNet to that of the source primate mapped to 251  
252 the same target primate. Therefore, as it stands, temporally-varying linear mappings to neu- 252  
253 ral responses collected from an animal during rapid visual stimulus presentation (RSVP) may 253  
254 not sufficiently separate feedforward models from recurrent models any better than one animal 254  
255 to another – though more investigation is needed to ensure tight estimates of the inter-animal 255  
256 consistency measure we have introduced here with neural data recorded from many primates. 256

257 Nonetheless, this observation motivates us to look beyond neural response predictions and turn 257  
258 to temporally-varying *behavioral* metrics in order to further separate these model classes, which 258  
259 we do next. 259

### 260 **2.3 ConvRNNs better match temporal dynamics of primate behavior than** 260 261 **feedforward models.** 261

262 To address whether recurrent processing is engaged during core object recognition behavior, we 262  
263 turn to behavioral data collected from behaving primates. There is a growing body of evidence 263  
264 that current feedforward models fail to accurately capture primate behavior<sup>30,17</sup>. We therefore 264  
265 reasoned that if recurrence is critical to core object recognition behavior, then recurrent net- 265  
266 works should be more consistent with suitable measures of primate behavior compared to the 266  
267 feedforward model family. Given that the identity of different objects is decoded from the IT 267  
268 population at different times, we considered the first time at which the IT neural decoding accu- 268  
269 racy reaches the (pooled) primate behavioral accuracy of a given image, known as the “object 269  
270 solution time (OST)”<sup>17</sup>. Given that our ConvRNNs also have an output at each *10ms* timebin, 270  
271 the procedure for computing the OST for these models is computed from its “IT-preferred” lay- 271  
272 ers, and we report the “OST consistency” which we define as the Spearman correlation between 272  
273 the model OSTs and the IT population’s OSTs on the common set of images solved by the given 273  
274 model and IT. 274

275 Unlike our ConvRNNs, which exhibit more biologically plausible temporal dynamics, eval- 275  
276 uating the temporal dynamics in feedforward models poses an immediate problem. Given that 276  
277 recurrent networks repeatedly apply nonlinear transformations across time, we can analogously 277  
278 map the layers of a feedforward network to timepoints, observing that a network with  $k$  dis- 278  
279 tinct layers can produce  $k$  distinct OSTs in this manner. Thus, the most direct definition of a 279  
280 feedforward model’s OST is to uniformly distribute the timebins between *70-260ms* across its  $k$  280



281 layers. For very deep feedforward networks such as ResNet-101 and ResNet-152, this number 281  
282 of distinct layers will be as fine-grained as the 10ms timebins of the IT responses; however, 282  
283 for most other shallower feedforward networks this will be much coarser. Therefore to enable 283  
284 shallow feedforward models to be maximally temporally expressive, we also randomly sample 284  
285 units from consecutive feedforward layers to produce a more graded temporal mapping, de- 285  
286 picted in Figure 4b. This graded mapping is ultimately what we use for the feedforward models 286  
287 in Figure 4d, providing the highest OST consistency for that model class<sup>d</sup>. 287

288 With model OST defined across both model families, we compared various ConvRNNs and 288  
289 feedforward models to the IT population's OST in Figure 4d. Among shallower and deeper 289  
290 models, we found that ConvRNNs were generally able to better explain IT's OST than their 290  
291 feedforward counterparts. Specifically, we found that ConvRNN cells without *any* multi-unit 291  
292 interaction such as the Time Decay ConvRNN only marginally, and not always significantly, 292  
293 improved the OST consistency over its respective BaseNet model<sup>e</sup>. On the other hand, consis- 293  
294 tent with our prior observation in Figure 3, ConvRNNs with multi-unit interactions generally 294  
295 provided the greatest match to IT OSTs than even the deepest feedforward models<sup>f</sup>. 295

296 Consistent with our observations in Figures 2 and 3 that different recurrent cells with multi- 296  
297 unit interactions were not all equally effective when embedded in CNNs (despite outperforming 297  
298 the simple Time Decay model), we similarly found that this observation held for the case of 298  
299 matching IT's OST. Given recent observations<sup>31</sup> that inactivating parts of macaque ventrolat- 299  
300 eral PFC (vlPFC) results in behavioral deficits in IT for late-solved images, we reasoned that 300  
301 additional decoding procedures employed at the categorization layer during task optimization 301

---

<sup>d</sup>Wilcoxon test on uniform vs. graded mapping OST consistencies across feedforward models,  $p < 0.001$ ; see also Figure S5.

<sup>e</sup>Paired  $t$ -test with Bonferroni correction: Shallow Time Decay vs. "BaseNet" in blue,  $t(9) \approx 3.23, p < 0.025$ ; Deeper Time Decay vs. "BaseNet" in red,  $t(9) \approx 1.73, p \approx 0.11$ .

<sup>f</sup>Paired  $t$ -test with Bonferroni correction: Shallow RGC vs. "BaseNet" in blue,  $t(9) \approx 6.08, p < 0.001$ ; Deeper UGRNN vs. ResNet-152,  $t(9) \approx 7.55, p < 0.001$ ; Deeper GRU vs. ResNet-152,  $t(9) \approx 7.71, p < 0.001$ ; RGC Median vs. ResNet-152,  $t(9) \approx 3.44, p < 0.01$ .

302 might meaningfully impact the model’s OST consistency, in addition to the choice of recur- 302  
303 rent cell used. We designed several decoding procedures (defined in Section A.5), motivated 303  
304 by prior observations of accumulation of relevant sensory signals during decision making in 304  
305 primates<sup>32</sup>. Overall, we found that ConvRNNs with different decoding procedures, but with 305  
306 the *same* layer-local recurrent cell (RGC Median) and long-range feedback connectivity pat- 306  
307 terns, yielded significant differences in final consistency with the IT population OST (Friedman 307  
308 test,  $p < 0.05$ ). Moreover, the simplest decoding procedure of outputting a prediction at the 308  
309 last timepoint, a strategy commonly employed by the computer vision community, had a lower 309  
310 OST consistency than each of the more nuanced Max Confidence<sup>g</sup> and Threshold decoding 310  
311 procedures<sup>h</sup> that we considered. Taken together, our results suggest that the type of multi-unit 311  
312 layer-wise recurrence *and* downstream decoding strategy are important features for OST con- 312  
313 sistency with IT, suggesting that specific, non-trivial connectivity patterns further downstream 313  
314 the ventral stream may be important to core object recognition behavior over timescales of a 314  
315 couple hundred milliseconds. 315

## 316 **2.4 ConvRNNs mediate a tradeoff between task performance and net-** 316 317 **work size.** 317

318 Why might a suitably shallower feedforward network with temporal dynamics be desirable for 318  
319 the ventral visual stream? We reasoned that recurrence mediates a tradeoff between network 319  
320 size and task performance; a tradeoff that the ventral stream also maintains. To examine this 320  
321 possibility, in Figure 5, we compared each network’s task performance versus its size, mea- 321  
322 sured either by parameter count or unit count. Across models, we found unit count (related 322  
323 to the number of neurons) to be more consistent with task performance than parameter count 323  
324 (related to the number of synapses). In fact, there are many models with a large parameter 324

---

<sup>g</sup>Paired  $t$ -test with Bonferroni correction,  $t(9) \approx -4.52$ ,  $p < 0.01$ .

<sup>h</sup>Paired  $t$ -test with Bonferroni correction,  $t(9) \approx -4.41$ ,  $p < 0.01$ .

325 count but not very good task performance, indicating that adding synapses is not necessarily as 325  
326 useful for performance as adding neurons. For shallower recurrent networks, task performance 326  
327 seemed to be more strongly associated with OST consistency than network size, though gener- 327  
328 ally having fewer parameters at a given performance level resulted in higher OST consistency 328  
329 (e.g. UGRNN vs. SimpleRNN and RGC vs. IntersectionRNN). This tradeoff became more 329  
330 salient for deeper feedforward models and the deeper ConvRNNs, as the very deep ResNets 330  
331 (ResNet-34 and deeper) attained an overall *lower* OST consistency compared to the deeper 331  
332 ConvRNNs, using both much more units and parameters compared to small relative gains in 332  
333 task performance. Similarly, deeper ConvRNNs with high task performance and minimal unit 333  
334 count, such as the UGRNN, GRU, and RGCs attained both the highest OST consistency overall 334  
335 (Figures 4 and 5) along with providing the best match to neural dynamics across visual areas 335  
336 (Figure 3). This observation indicates that suitably-chosen recurrence can provide a means for 336  
337 maintaining this fundamental tradeoff. 337

338 Given that specific forms of task-optimized recurrence are more consistent with IT's OST 338  
339 than iterated feedforward transformations (with unshared weights), we asked whether it was 339  
340 possible to approximate the effect of recurrence with a feedforward model. This approxima- 340  
341 tion would allow us to better describe the additional “action” that recurrence is providing in 341  
342 its improved OST consistency. Furthermore, a crucial difference between this metric and the 342  
343 explained variance metric evaluated on neural data in the prior section is that the latter uses a 343  
344 linear transform from model features to neural responses, whereas the former operates directly 344  
345 on the original model features. Therefore, a related question is whether the use of a linear trans- 345  
346 form for mapping from model units to neural responses *masks* the improvement that recurrent 346  
347 processing can have over deep feedforward models in their original feature space. 347

348 To address these questions, we trained a separate linear mapping from each model layer to 348  
349 the corresponding IT response at the given timepoint, on a set of images distinct from those on 349

350 which the OST consistency metric is evaluated on. Overall, as depicted in Figure S5, we found 350  
351 that models with *less* temporal variation in their source features (namely those under a uniform 351  
352 mapping with few “IT-preferred” layers) had significantly *improved* OST consistency with their 352  
353 linearly transformed features (Wilcoxon test,  $p < 0.001$ ), whereas models with the maximum 353  
354 amount of temporal variation such as ResNet-101, ResNet-152, and the ConvRNNs had a sig- 354  
355 nificant reduction in OST consistency with their linearly transformed features (Wilcoxon test, 355  
356  $p < 0.001$ ), indicating the harmful dimensionality reduction of the linear mapping. On the 356  
357 other hand, the linearly transformed shallower variants of deeper feedforward models were *not* 357  
358 significantly different from task-optimized ConvRNNs that achieved high OST consistency<sup>i</sup>, 358  
359 suggesting that the action of suitable task-optimized recurrence approximates that of a shal- 359  
360 lower feedforward model with linearly induced neural dynamics. 360

## 361 Discussion 361

362 The overall goal of this study is to determine what role recurrent circuits may have in the ex- 362  
363 ecution of core object recognition behavior in the ventral stream. By broadening the method 363  
364 of goal-driven modeling from solving tasks with feedforward CNNs to ConvRNNs that include 364  
365 layer-local recurrence and feedback connections, we first demonstrate that appropriate choices 365  
366 of these recurrent circuits which incorporate specific principles of “gating” and “bypassing” 366  
367 lead to matching the task performance of much deeper feedforward CNNs with fewer units 367  
368 and parameters. Moreover, unlike deep feedforward CNNs, the mapping from the early, in- 368  
369 termediate, and higher layers of these shallower ConvRNNs to corresponding cortical areas is 369  
370 neuroanatomically consistent and reproduces prior quantitative properties of the ventral stream. 370  
371 We further find that these task-optimized ConvRNNs can reliably produce dynamic neural re- 371

---

<sup>i</sup>Paired  $t$ -test with Bonferroni correction: RGC Median vs. PLS Uniform BaseNet,  $t(9) \approx -0.86, p \approx 0.41$ ; RGC Median with Threshold Decoder vs. PLS Uniform ResNet-18,  $t(9) \approx 0.82, p \approx 0.43$ ; RGC Median with Max Confidence Decoder vs. PLS Uniform ResNet-34,  $t(9) \approx 0.02, p \approx 0.99$ .

372 sponse trajectories at temporal resolutions of tens of milliseconds throughout the ventral visual 372  
373 hierarchy, unlike feedforward models or certain other choices for recurrence adapted to solving 373  
374 visual recognition problems such as Temporal Decay or the LSTM circuit. 374

375 In fact, ConvRNNs with high task performance but small network size (as measured by 375  
376 number of neurons rather than synapses) are not only the most quantitatively accurate models 376  
377 of neural response trajectories during passive viewing but also are most consistent with the tem- 377  
378 poral evolution of primate IT object identity solutions during active task performance. Taken 378  
379 together, our results suggest that recurrence in the ventral stream mediates a tradeoff between 379  
380 task performance and neuron count, suggesting that the computer vision community's solution 380  
381 of stacking more feedforward layers to solve challenging visual recognition problems approx- 381  
382 imates what is compactly implemented in the primate visual system by leveraging additional 382  
383 nonlinear temporal transformations to the initial feedforward IT response. This work therefore 383  
384 provides a quantitative prescription for the next generation of dynamic ventral stream mod- 384  
385 els, addressing the call to action in a recent previous study<sup>17</sup> for a change in architecture from 385  
386 feedforward models. 386

387 Many hypotheses about the role of recurrence in vision have been put forward, particu- 387  
388 larly in regards to overcoming certain challenging image properties<sup>10,11,12,13,9,14,15,16,17,18,19,20</sup>. 388  
389 We believe this is the first work to address the role of recurrence at scale by connecting novel 389  
390 *task-optimized* recurrent models to temporal metrics defined on high-throughput neural and be- 390  
391 havioral data. Moreover, these metrics are well-defined for feedforward models (unlike prior 391  
392 work<sup>33</sup>) and therefore meaningfully demonstrate a separation between the two model classes. 392

393 Though our results help to clarify the role of recurrence during core object recognition be- 393  
394 havior, many major questions remain. Our work addresses why the visual system may leverage 394  
395 recurrence to subservise visually challenging behaviors, replacing a physically implausible archi- 395  
396 tecture (deep feedforward CNNs) with one that is ubiquitously consistent with anatomical ob- 396

397 servations (shallower ConvRNNs). However, our work does not address gaps in understanding 397  
398 either the loss function or the learning rule of the neural network. Specifically, we intentionally 398  
399 implant layer-local recurrence and long-range feedback connections into feedforward networks 399  
400 that have been useful for supervised learning via backpropagation on ImageNet. A natural next 400  
401 step would be to connect these ConvRNNs with unsupervised objectives, as has been done for 401  
402 feedforward models of the ventral stream in concurrent work<sup>34</sup>. The question of biologically 402  
403 plausible learning targets is similarly linked to biologically plausible mechanisms for learning 403  
404 such objective functions. Recurrence could play a separate role in implementing the propaga- 404  
405 tion of error-driven learning, obviating the need for some of the issues with backpropagation 405  
406 (such as weight transport), as has been recently demonstrated at scale<sup>35,36</sup>. Therefore, building 406  
407 ConvRNNs with unsupervised objective functions optimized with biologically-plausible learn- 407  
408 ing rules would be essential towards a more complete goal-driven theory of visual cortex. 408

409 Additionally, high-throughput experimental data will also be critical to further separate hy- 409  
410 potheses about recurrence. While we see evidence of recurrence as mediating a tradeoff between 410  
411 network size and task performance for core object recognition, it could be that recurrence plays 411  
412 a more task-specific role under more temporally dynamic behaviors. Not only would it be an 412  
413 interesting direction to optimize ConvRNNs on more temporally dynamic visual tasks than Im- 413  
414 ageNet, but to compare to neural and behavioral data collected from such stimuli, potentially 414  
415 over longer timescales than *200ms* while the animal is performing a task. Such models and 415  
416 experimental data would synergistically provide great insight into how rich visual behaviors 416  
417 proceed, while also inspiring better computer vision algorithms. 417

## 418 **References** 418

419 [1] James, W. The principles of psychology (vol. 1). *New York: Holt* **474** (1890). 419

- 420 [2] Pinto, N., Cox, D. D. & Dicarlo, J. J. Why is real-world visual object recognition hard? 420  
421 *PLoS Computational Biology* (2008). 421
- 422 [3] DiCarlo, J. J., Zoccolan, D. & Rust, N. C. How does the brain solve visual object recog- 422  
423 nition? *Neuron* **73**, 415–34 (2012). 423
- 424 [4] Yamins, D. L. K. *et al.* Performance-optimized hierarchical models predict neural re- 424  
425 sponses in higher visual cortex. *Proceedings of the National Academy of Sciences* **111**, 425  
426 8619–8624 (2014). [http://www.pnas.org/content/111/23/8619.full.](http://www.pnas.org/content/111/23/8619.full.pdf) 426  
427 pdf. 427
- 428 [5] Khaligh-Razavi, S.-M. & Kriegeskorte, N. Deep supervised, but not unsupervised, models 428  
429 may explain it cortical representation. *PLoS computational biology* **10**, e1003915 (2014). 429
- 430 [6] Güçlü, U. & van Gerven, M. A. Deep neural networks reveal a gradient in the complex- 430  
431 ity of neural representations across the ventral stream. *The Journal of Neuroscience* **35**, 431  
432 10005–10014 (2015). 432
- 433 [7] Yamins, D. L. & DiCarlo, J. J. Using goal-driven deep learning models to understand 433  
434 sensory cortex. *Nature neuroscience* **19**, 356 (2016). 434
- 435 [8] Cadena, S. A. *et al.* Deep convolutional models improve predictions of macaque v1 re- 435  
436 sponses to natural images. *PLoS computational biology* **15**, e1006897 (2019). 436
- 437 [9] Gilbert, C. D. & Wu, L. Top-down influences on visual processing. *Nat. Rev. Neurosci.* 437  
438 **14**, 350–363 (2013). 438
- 439 [10] Spoerer, C. J., McClure, P. & Kriegeskorte, N. Recurrent convolutional neural networks: 439  
440 a better model of biological object recognition. *Front. Psychol.* **8**, 1–14 (2017). 440

- 441 [11] Michaelis, C., Bethge, M. & Ecker, A. One-shot segmentation in clutter. In *International* 441  
442 *Conference on Machine Learning*, 3549–3558 (PMLR, 2018). 442
- 443 [12] Rajaei, K., Mohsenzadeh, Y., Ebrahimpour, R. & Khaligh-Razavi, S.-M. Beyond core 443  
444 object recognition: Recurrent processes account for object recognition under occlusion. 444  
445 *PLoS computational biology* **15**, e1007001 (2019). 445
- 446 [13] Linsley, D., Kim, J., Veerabadrán, V., Windolf, C. & Serre, T. Learning long- 446  
447 range spatial dependencies with horizontal gated recurrent units. In Bengio, S. *et al.* 447  
448 (eds.) *Advances in Neural Information Processing Systems*, vol. 31 (Curran Associates, 448  
449 Inc., 2018). URL [https://proceedings.neurips.cc/paper/2018/file/](https://proceedings.neurips.cc/paper/2018/file/ec8956637a99787bd197eacd77acce5e-Paper.pdf) 449  
450 [ec8956637a99787bd197eacd77acce5e-Paper.pdf](https://proceedings.neurips.cc/paper/2018/file/ec8956637a99787bd197eacd77acce5e-Paper.pdf). 450
- 451 [14] Lindsay, G. W. Feature-based attention in convolutional neural networks. *arXiv preprint* 451  
452 *arXiv:1511.06408* (2015). 452
- 453 [15] McIntosh, L., Maheswaranathan, N., Sussillo, D. & Shlens, J. Recurrent segmentation 453  
454 for variable computational budgets. In *Proceedings of the IEEE Conference on Computer* 454  
455 *Vision and Pattern Recognition Workshops*, 1648–1657 (2018). 455
- 456 [16] Li, X., Jie, Z., Feng, J., Liu, C. & Yan, S. Learning with rethinking: recurrently improving 456  
457 convolutional neural networks through feedback. *Pattern Recognition* **79**, 183–194 (2018). 457
- 458 [17] Kar, K., Kubilius, J., Schmidt, K., Issa, E. B. & DiCarlo, J. J. Evidence that recurrent 458  
459 circuits are critical to the ventral stream’s execution of core object recognition behavior. 459  
460 *Nature neuroscience* **22**, 974–983 (2019). 460
- 461 [18] Rao, R. P. & Ballard, D. H. Predictive coding in the visual cortex: a functional interpreta- 461  
462 tion of some extra-classical receptive-field effects. *Nature neuroscience* **2**, 79–87 (1999). 462



- 463 [19] Lotter, W., Kreiman, G. & Cox, D. Deep predictive coding networks for video prediction 463  
464 and unsupervised learning. In *ICLR (2017)*. 464
- 465 [20] Issa, E. B., Cadieu, C. F. & DiCarlo, J. J. Neural dynamics at successive stages of the 465  
466 ventral visual stream are consistent with hierarchical error signals. *Elife* **7**, e42870 (2018). 466
- 467 [21] Liao, Q. & Poggio, T. Bridging the gaps between residual learning, recurrent neural 467  
468 networks and visual cortex. *arXiv preprint arXiv:1604.03640* (2016). 468
- 469 [22] Zamir, A. R. *et al.* Feedback networks. In *CVPR (2017)*. 469
- 470 [23] Leroux, S. *et al.* Iamnn: iterative and adaptive mobile neural network for efficient image 470  
471 classification. In *ICLR Workshop 2018 (2018)*. 471
- 472 [24] Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation 472  
473 by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013). 473
- 474 [25] Collins, J., Sohl-Dickstein, J. & Sussillo, D. Capacity and trainability in recurrent neural 474  
475 networks. In *ICLR (2017)*. 475
- 476 [26] Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolu- 476  
477 tional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L. & Weinberger, K. Q. 477  
478 (eds.) *Advances in Neural Information Processing Systems*, vol. 25 (Curran Associates, 478  
479 Inc., 2012). URL [https://proceedings.neurips.cc/paper/2012/file/](https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf) 479  
480 [c399862d3b9d6b76c8436e924a68c45b-Paper.pdf](https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf). 480
- 481 [27] Nayebi, A. *et al.* Task-driven convolutional recurrent models of the visual system. In 481  
482 Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 31 (Cur- 482  
483 ran Associates, Inc., 2018). URL [https://proceedings.neurips.cc/paper/](https://proceedings.neurips.cc/paper/2018/file/6be93f7a96fed60c477d30ae1de032fd-Paper.pdf) 483  
484 [2018/file/6be93f7a96fed60c477d30ae1de032fd-Paper.pdf](https://proceedings.neurips.cc/paper/2018/file/6be93f7a96fed60c477d30ae1de032fd-Paper.pdf). 484

- 485 [28] Kietzmann, T. C. *et al.* Recurrence is required to capture the representational dynamics of 485  
486 the human visual system. *Proceedings of the National Academy of Sciences* **116**, 21854– 486  
487 21863 (2019). 487
- 488 [29] Majaj, N. J., Hong, H., Solomon, E. A. & DiCarlo, J. J. Simple learned weighted sums 488  
489 of inferior temporal neuronal firing rates accurately predict human core object recognition 489  
490 performance. *Journal of Neuroscience* **35**, 13402–13418 (2015). 490
- 491 [30] Rajalingham, R. *et al.* Large-scale, high-resolution comparison of the core visual ob- 491  
492 ject recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural 492  
493 networks. *Journal of Neuroscience* **38**, 7255–7269 (2018). 493
- 494 [31] Kar, K. & DiCarlo, J. J. Fast recurrent processing via ventrolateral prefrontal cortex is 494  
495 needed by the primate ventral stream for robust core visual object recognition. *Neuron* 495  
496 **109**, 164–176 (2021). 496
- 497 [32] Shadlen, M. N. & Newsome, W. T. Neural basis of a perceptual decision in the parietal 497  
498 cortex (area lip) of the rhesus monkey. *Journal of neurophysiology* **86**, 1916–1936 (2001). 498
- 499 [33] Kumbhani, J. *et al.* Brain-like object recognition with high-performing shallow recurrent 499  
500 units. In Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems*, 500  
501 vol. 32 (Curran Associates, Inc., 2019). URL [https://proceedings.neurips.](https://proceedings.neurips.cc/paper/2019/file/7813d1590d28a7dd372ad54b5d29d033-Paper.pdf) 501  
502 [cc/paper/2019/file/7813d1590d28a7dd372ad54b5d29d033-Paper.](https://proceedings.neurips.cc/paper/2019/file/7813d1590d28a7dd372ad54b5d29d033-Paper.pdf) 502  
503 [pdf.](https://proceedings.neurips.cc/paper/2019/file/7813d1590d28a7dd372ad54b5d29d033-Paper.pdf) 503
- 504 [34] Zhuang, C. *et al.* Unsupervised neural network models of the ventral visual stream. *Pro-* 504  
505 *ceedings of the National Academy of Sciences* **118** (2021). 505
- 506 [35] Akrouf, M., Wilson, C., Humphreys, P., Lillicrap, T. & Tweed, D. B. Deep 506  
507 learning without weight transport. In Wallach, H. *et al.* (eds.) *Advances* 507

- 508        *in Neural Information Processing Systems*, vol. 32 (Curran Associates, Inc., 508  
509        2019).        URL [https://proceedings.neurips.cc/paper/2019/file/](https://proceedings.neurips.cc/paper/2019/file/f387624df552cea2f369918c5e1e12bc-Paper.pdf) 509  
510        [f387624df552cea2f369918c5e1e12bc-Paper.pdf](https://proceedings.neurips.cc/paper/2019/file/f387624df552cea2f369918c5e1e12bc-Paper.pdf). 510
- 511 [36] Kunin, D. *et al.* Two routes to scalable credit assignment without weight symmetry. In 511  
512        *International Conference on Machine Learning*, 5511–5521 (PMLR, 2020). 512
- 513 [37] Abadi, M. *et al.* Tensorflow: A system for large-scale machine learning. In *OSDI*, vol. 16, 513  
514        265–283 (2016). 514
- 515 [38] Mizuseki, K., Sirota, A., Pastalkova, E. & Buzsáki, G. Theta oscillations provide temporal 515  
516        windows for local circuit computation in the entorhinal-hippocampal loop. *Neuron* 267– 516  
517        280 (2009). 517
- 518 [39] Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the importance of initialization and 518  
519        momentum in deep learning. In *International conference on machine learning*, 1139–1147 519  
520        (PMLR, 2013). 520
- 521 [40] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In 521  
522        *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 522  
523        (2016). 523
- 524 [41] Clevert, D.-A., Unterthiner, T. & Hochreiter, S. Fast and accurate deep network learning 524  
525        by exponential linear units (elus). In *ICLR* (2016). 525
- 526 [42] Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by 526  
527        reducing internal covariate shift. In *International Conference on Machine Learning*, 448– 527  
528        456 (PMLR, 2015). 528

- 529 [43] Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient 529  
530 descent is difficult. *IEEE transactions on neural networks* **5**, 157–166 (1994). 530
- 531 [44] Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer normalization. *arXiv preprint* 531  
532 *arXiv:1607.06450* (2016). 532
- 533 [45] Cho, K., van Merriënboer, B., Bahdanau, D. & Bengio, Y. On the properties of neural ma- 533  
534 chine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Work-* 534  
535 *shop on Syntax, Semantics and Structure in Statistical Translation*, 103–111 (Association 535  
536 for Computational Linguistics, Doha, Qatar, 2014). URL [https://www.aclweb.](https://www.aclweb.org/anthology/W14-4012) 536  
537 [org/anthology/W14-4012](https://www.aclweb.org/anthology/W14-4012). 537
- 538 [46] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation* **9**, 1735– 538  
539 1780 (1997). 539
- 540 [47] Jozefowicz, R., Zaremba, W. & Sutskever, I. An empirical exploration of recurrent 540  
541 network architectures. In *International Conference on Machine Learning*, 2342–2350 541  
542 (PMLR, 2015). 542
- 543 [48] Gers, F. A., Schraudolph, N. N. & Schmidhuber, J. Learning precise timing with lstm 543  
544 recurrent networks. *Journal of machine learning research* **3**, 115–143 (2002). 544
- 545 [49] Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. Algorithms for hyper-parameter opti- 545  
546 mization. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F. & Weinberger, K. Q. 546  
547 (eds.) *Advances in Neural Information Processing Systems*, vol. 24 (Curran Associates, 547  
548 Inc., 2011). URL [https://proceedings.neurips.cc/paper/2011/file/](https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf) 548  
549 [86e8f7ab32cfd12577bc2619bc635690-Paper.pdf](https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf). 549
- 550 [50] Bergstra, J., Komer, B., Eliasmith, C., Yamins, D. & Cox, D. D. Hyperopt: a python 550

- 551 library for model selection and hyperparameter optimization. *Computational Science &* 551  
552 *Discovery* **8** (2015). 552
- 553 [51] He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human- 553  
554 level performance on imagenet classification. In *Proceedings of the IEEE international* 554  
555 *conference on computer vision*, 1026–1034 (2015). 555
- 556 [52] Nesterov, Y. A method of solving a convex programming problem with convergence rate 556  
557  $o(1/k^2)$ . In *Sov. Math. Dokl*, vol. 27. 557
- 558 [53] Klindt, D., Ecker, A. S., Euler, T. & Bethge, M. Neural system identification 558  
559 for large populations separating “what” and “where”. In Guyon, I. *et al.* (eds.) 559  
560 *Advances in Neural Information Processing Systems*, vol. 30 (Curran Associates, 560  
561 Inc., 2017). URL [https://proceedings.neurips.cc/paper/2017/file/](https://proceedings.neurips.cc/paper/2017/file/8c249675aea6c3cbd91661bbae767ff1-Paper.pdf) 561  
562 [8c249675aea6c3cbd91661bbae767ff1-Paper.pdf](https://proceedings.neurips.cc/paper/2017/file/8c249675aea6c3cbd91661bbae767ff1-Paper.pdf). 562
- 563 [54] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *ICLR* (2015). 563
- 564 [55] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a 564  
565 simple way to prevent neural networks from overfitting. *The journal of machine learning* 565  
566 *research* **15**, 1929–1958 (2014). 566

## 567 **Acknowledgements**

567

568 We thank Tyler Bonnen and Eshed Margalit for comments on this manuscript. We thank the  
569 Google TensorFlow Research Cloud (TFRC) team for generously providing TPU hardware re-  
570 sources for this project. D.L.K.Y is supported by the James S. McDonnell Foundation (Un-  
571 derstanding Human Cognition Award Grant No. 220020469), the Simons Foundation (Collab-  
572 oration on the Global Brain Grant No. 543061), the Sloan Foundation (Fellowship FG-2018-  
573 10963), the National Science Foundation (RI 1703161 and CAREER Award 1844724), the  
574 DARPA Machine Common Sense program, and hardware donation from the NVIDIA Corpora-  
575 tion. This work is also supported in part by Simons Foundation grant SCGB-542965 (J.J.D. &  
576 D.L.K.Y.). This project has received funding from the European Union’s Horizon 2020 research  
577 and innovation programme under grant agreement No. 70549 (J.K.). J.S. is supported by the  
578 Mexican National Council of Science and Technology (CONACYT). 578

## 579 **Author Contributions**

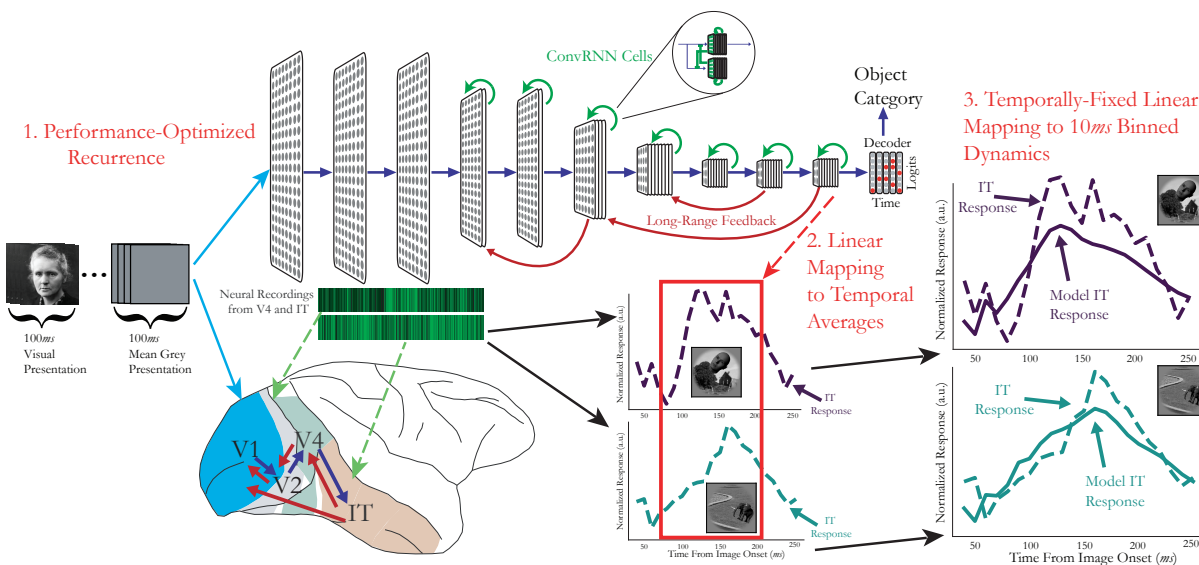
579

580 A.N. and D.L.K.Y. designed the experiments. A.N., J.S., and D.B. conducted the experiments,  
581 and A.N. analyzed the data. K.K. contributed neural data, and J.K. contributed to initial code  
582 development. K.K. and J.J.D. provided technical advice on neural predictivity metrics. D.S.  
583 and S.G. provided technical advice on recurrent neural network training. A.N. and D.L.K.Y.  
584 interpreted the data and wrote the paper. 584

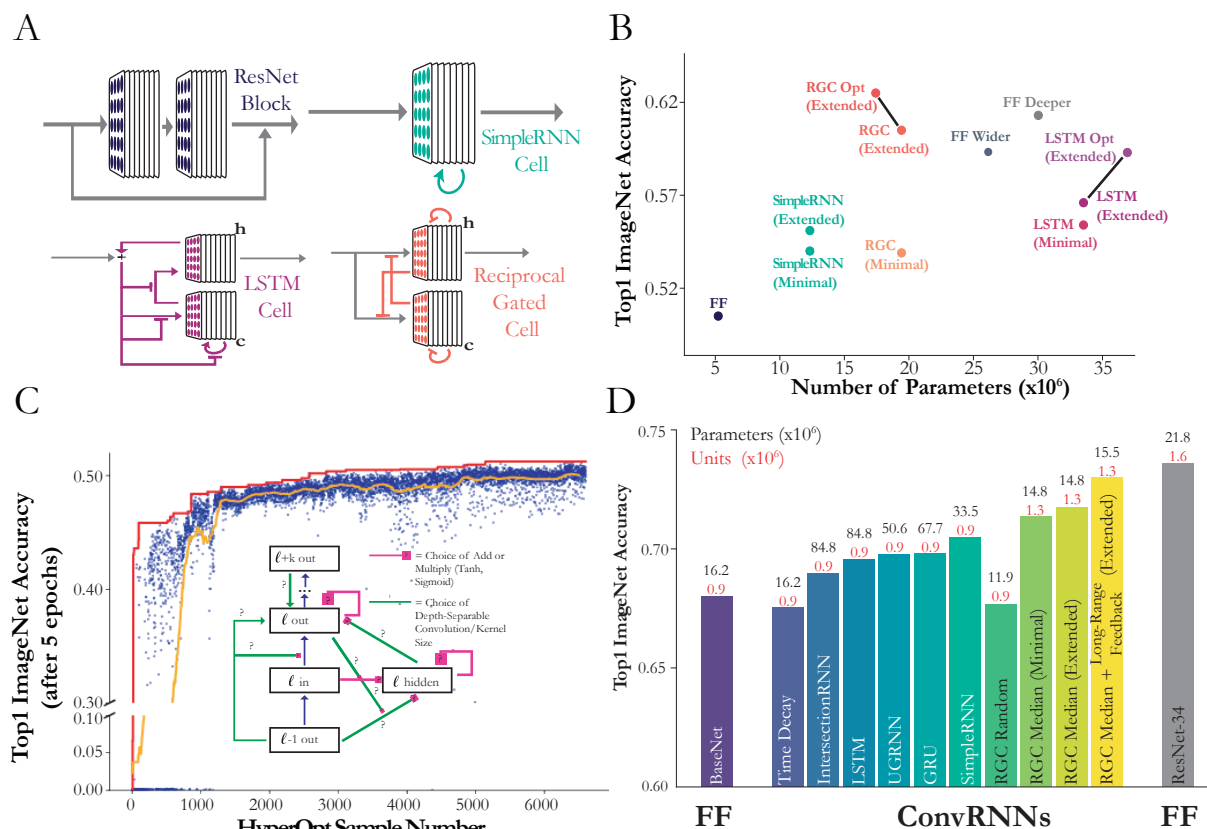
## 585 **Competing Interest Declaration**

585

586 The authors declare no competing interests. 586

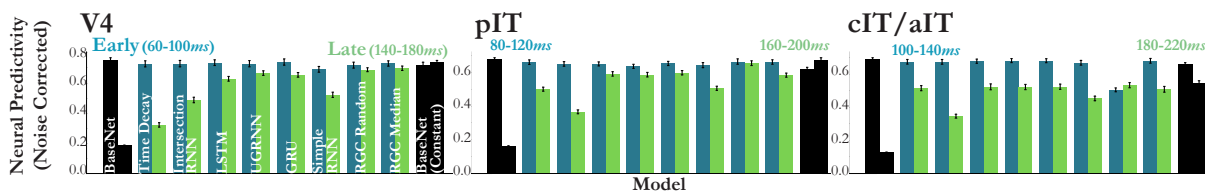


**Figure 1: ConvRNNs as models of the primate ventral visual stream.** *Performance-optimized recurrence.* Convolutional recurrent networks (ConvRNNs) have a combination of local recurrent cells (green) and long-range feedback connections (red) added on top of a feedforward CNN “BaseNet” backbone (blue). In our implementation displayed on the top, propagation along each arrow takes one time step (10ms) to mimic conduction delays between cortical layers. In addition, we consider particular choices of “light-weight” (in terms of parameter count) decoding strategy that determines the final object category of that image. *Linear mapping to temporal averages.* We stipulated that units from each multi-unit array must be fit by features from a single model layer. To determine which one, we fit the features from the feedforward backbone to the unit’s time-averaged response, and counted how many units had minimal loss for a given model layer, detailed in Section A.6.2. *Temporally-fixed linear mapping to 10ms binned dynamics.* The ConvRNN model features produce a temporally-varying output that is mapped linearly to temporally-varying neural responses in V4 and IT, under a temporally-fixed mapping whose parameters are reused throughout the entire timecourse of stimulus presentation.

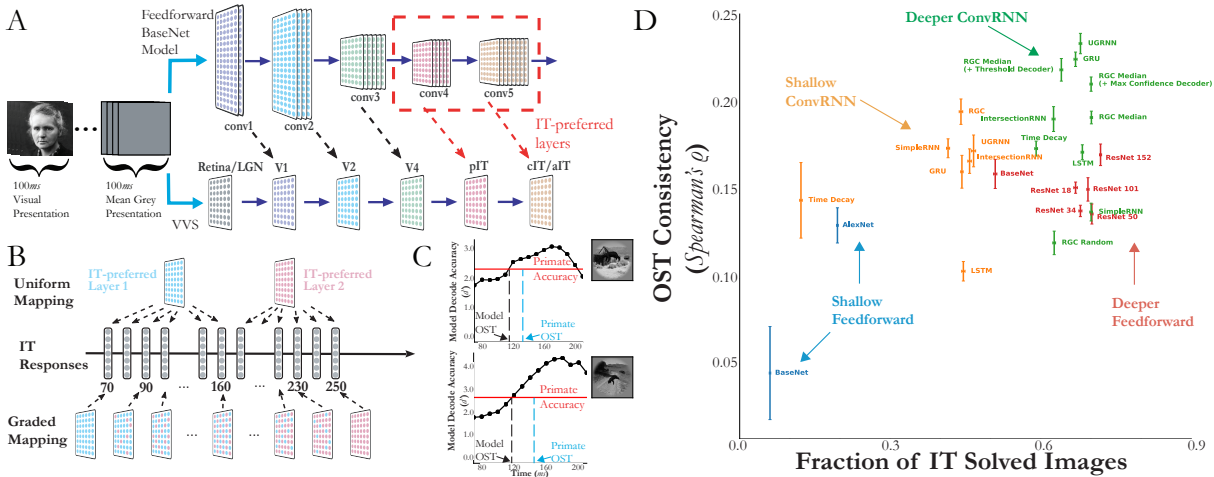


**Figure 2: Suitably-chosen ConvRNNs can match the object recognition performance of much deeper feedforward models. (a) Architectural differences between ConvRNN cells.** Standard ResNet blocks and SimpleRNN cells have bypassing but no gating. The LSTM cell has gating, denoted by T-junctions, but no bypassing. The Reciprocal Gated Cell has both. **(b) Performance of various ConvRNN and feedforward models as a function of number of parameters.** Colored points incorporate the respective ConvRNN cell into the the 6-layer feedforward BaseNet architecture (“FF”). Here “T” denotes number of timesteps the model is unrolled for, corresponding to the propagation of a single layer to the next. Hyperparameter-optimized versions of the LSTM (“LSTM Opt”) and Reciprocal Gated Cell ConvRNNs (“RGC Opt”) are connected to their non-optimized versions by black lines. **(c) ConvRNN cell search.** Each blue dot represents a model, sampled from hyperparameter space, trained for 5 epochs. The orange line is the average performance of the last 50 models up to that time. The red line denotes the top performing model at that point in the search. *Search space schematic:* Question marks denote optional connections, which may be conventional or depth-separable convolutions with a choice of kernel size. **(d) Performance of models fully trained on ImageNet.** We compared the performance of an 11-layer feedforward base model (“BaseNet”) modeled after ResNet-18, a control ConvRNN model with trainable time constants (“Time Decay”), along with various other common RNN architectures implanted into this BaseNet, as well as the median Reciprocal Gated Cell (RGC) model from the search (“RGC Median”) with or without global feedback connectivity, and its minimally-unrolled control ( $T = 12$ ). The “RGC Random” model was selected randomly from the initial, random phase of the model search. Parameter and unit counts (total number of neurons in the output of each layer) in millions are shown on top of each bar.

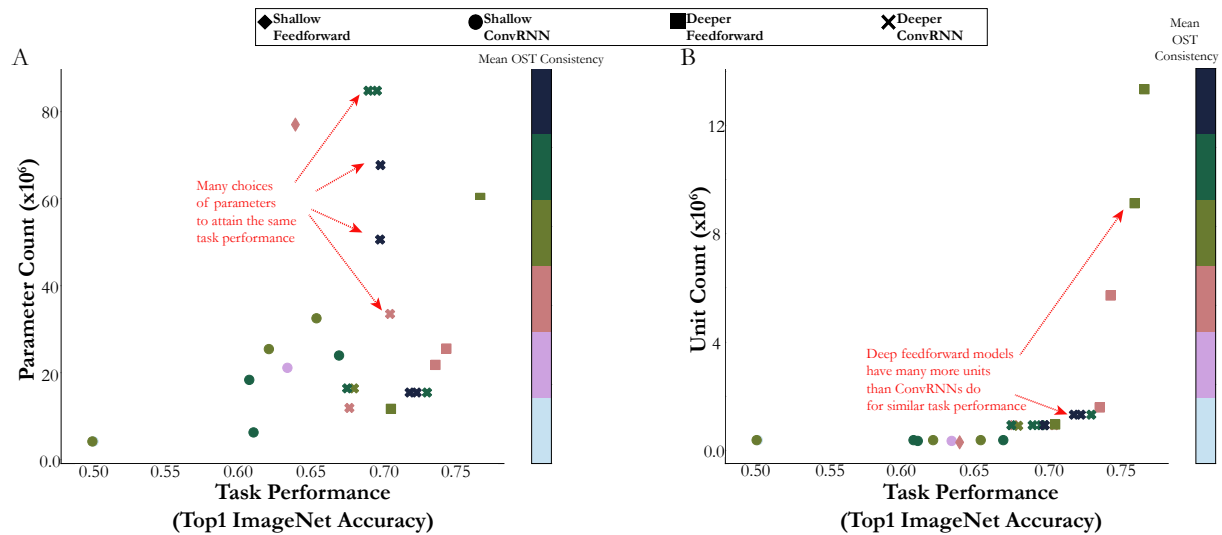




**Figure 3: Suitably-chosen ConvRNN circuits provide consistent predictions of primate ventral stream neural dynamics.** The  $y$ -axis indicates the median across neurons of the explained variance between predictions and ground-truth responses on held-out images divided by the square root of the internal consistencies of the neurons, defined in Section A.6.3. Error bars indicates the s.e.m across neurons ( $N = 88$  for V4,  $N = 88$  for pIT,  $N = 80$  for cIT/aIT) averaged across  $10ms$  timebins ( $N = 4$  each for the “Early” and “Late” designations). As can be seen, the feedforward BaseNet model (first bars) is incapable of generating a response beyond the feedforward pass, but certain types of ConvRNN cells (such as “RGC Median”, “UGRNN”, and “GRU”) added to the feedforward model are overall best predictive across visual areas at late timepoints (Wilcoxon test (with Bonferroni correction) with feedforward BaseNet,  $p < 0.001$  for each visual area). “BaseNet (Constant)” refers to the same feedforward model but presented with a constant image presentation, in contrast to the other models which are given the same  $100ms$  stimulus presentation as the primate. See Figure S3 for the full timecourses at the resolution of  $10ms$  bins.



**Figure 4: ConvRNNs explain the object solution times (OST) of IT across images.** (a) **Mapping model layers to timepoints.** In order to compare to primate IT object solution times, namely the first time at which the neural decode accuracy for each image reached the level of the (pooled) primate behavioral accuracy, we first need to define object solution times for models. This procedure involves identification of the “IT-preferred” layer(s) via a standard linear mapping to temporally averaged IT responses. (b) **Choosing a temporal mapping gradation.** These “IT-preferred” model layer(s) are then mapped to 10ms timebins from 70-260ms in either a uniform or graded fashion, if the model is feedforward. For ConvRNNs, this temporal mapping is always one-to-one with these 10ms timebins. (c) **Defining model OSTs.** Once the temporal mapping has been defined, we train a linear SVM at each 10ms model timebin and compute the classifier’s  $d'$  (displayed in each of the black dots for a given example image). The first timebin at which the model  $d'$  matches the primate’s accuracy is defined as the model OST for that image (obtained via linear interpolation). (d) **Proper choices of recurrence best match IT OSTs.** Mean and s.e.m. are computed across train/test splits ( $N = 10$ ) when that image (of 1320 images) was a test-set image, with the Spearman correlation computed with the IT object solution times (analogously computed from the IT population responses) across the imageset solved by both the given model and IT, constituting the “Fraction of IT Solved Images” on the  $x$ -axis. We start with either a shallower base feedforward model consisting of 5 convolutional layers and 1 layer of readout (“BaseNet” in blue) as well as a deeper variant with 10 feedforward layers and 1 layer of readout (“BaseNet” in red), detailed in Section A.2.1. From these base feedforward models, we embed recurrent cells, resulting in either “Shallow ConvRNNs” or “Deeper ConvRNNs”.



**Figure 5: ConvRNNs with highest OST consistency conserve on network size while maintaining task performance.** Across all models considered, the Deeper ConvRNNs (denoted by “x”) that attain high categorization performance ( $x$ -axis) while maintaining a low unit count (panel B) rather than parameter count (panel A) for their given performance level, achieve the highest mean OST consistency (Spearman correlation with IT population OST, averaged across  $N = 10$  train/test splits). The colorbar indicates this mean OST consistency (monotonically increasing from light to dark), binned into 6 equal ranges (0.04478253 – 0.0769489, 0.0769489 – 0.10911527, 0.10911527 – 0.14128164, 0.14128164 – 0.17344802, 0.17344802 – 0.20561439, and 0.20561439 – 0.23778076). Models with a larger size at a fixed performance level are *less consistent* with primate object recognition behavior (e.g. deep feedforward models, denoted by boxes), with recurrence maintaining a fundamental tradeoff between network size and task performance.

## 587 **A Methods** 587

### 588 **A.1 Model framework** 588

#### 589 **A.1.1 Software package** 589

590 To explore the architectural space of ConvRNNs and compare these models with the primate 590  
591 visual system, we used the Tensorflow library<sup>37</sup> to augment standard CNNs with both local 591  
592 and long-range recurrence (Figure 1). Conduction from one area to another in visual cortex 592  
593 takes approximately  $10ms$ <sup>38</sup>, with signal from photoreceptors reaching IT cortex at the top of 593  
594 the ventral stream by  $70-100ms$ . Neural dynamics indicating potential recurrent connections 594  
595 take place over the course of  $100-260ms$ <sup>20</sup>. A single feedforward volley of responses thus 595  
596 cannot be treated as if it were instantaneous relative to the timescale of recurrence and feedback. 596  
597 Hence, rather than treating each entire feedforward pass from input to output as one integral time 597  
598 step, as is normally done with RNNs<sup>10</sup>, each time step in our models corresponds to a single 598  
599 feedforward layer processing its input and passing it to the next layer. This choice required 599  
600 an unrolling scheme different from that used in the standard Tensorflow RNN library, the code 600  
601 for which (and for all of our models) can be found at in our TNN Github repository: `https :` 601  
602 `//github.com/neuroailab/tnn.` 602

#### 603 **A.1.2 Defining ConvRNNs** 603

604 Within each ConvRNN layer, feedback inputs from higher layers are resized to match the spatial 604  
605 dimensions of the feedforward input to that layer. Both types of input are processed by standard 605  
606 2-D convolutions. If there is any local recurrence at that layer, the output is next passed to the 606  
607 recurrent cell as input. Feedforward and feedback inputs are combined within the recurrent cell 607  
608 by spatially resizing the feedback inputs (via bilinear interpolation) and concatenating these 608  
609 with the feedforward input across the channel dimension. We let  $\oplus$  denote this concatenation 609  
610 along the channel dimension with appropriate resizing to align spatial dimensions. Finally, the 610

611 output of the cell is passed through any additional nonlinearities, such as max-pooling. The 611  
612 generic update rule for the discrete-time trajectory of such a network is thus: 612

$$\begin{aligned} h_t^\ell &= C_\ell \left( F_\ell \left( \bigoplus_{j \neq \ell} r_t^j \right), h_{t-1}^\ell \right) \\ r_t^\ell &= A_\ell(h_t^\ell), \end{aligned} \quad (1)$$

613 where  $r_t^\ell$  is the output of layer  $\ell$  at time  $t$ ,  $h_{t-1}^\ell$  is the hidden state of the locally recurrent cell  $C_\ell$  613  
614 at time  $t - 1$ , and  $A_\ell$  is the activation function and any pooling post-memory operations. The 614  
615 learned parameters of such a network consist of  $F_\ell$ , comprising any feedforward and feedback 615  
616 connections coming into layer  $\ell = 1, \dots, L$ , and any of the learned parameters associated with 616  
617 the local recurrent cell  $C_\ell$ . 617

618 In this work, all forms of recurrence add parameters to the feedforward base model. Because 618  
619 this could improve task performance for reasons unrelated to recurrent computation, we trained 619  
620 two types of control model to compare to ConvRNNs: 620

- 621 1. Feedforward models with more convolution filters (“wider”) or more layers (“deeper”) to 621  
622 approximately match the number of parameters in a recurrent model. 622
- 623 2. Replicas of each ConvRNN model unrolled for a *minimal* number of time steps, defined 623  
624 as the number that allows all model parameters to be used at least once. A minimally un- 624  
625 rolled model has exactly the same number of parameters as its fully unrolled counterpart, 625  
626 so any increase in performance from unrolling longer can be attributed to recurrent com- 626  
627 putation. Fully and minimally unrolled ConvRNNs were trained with identical learning 627  
628 hyperparameters. 628

### 629 **A.1.3 Training Procedure** 629

630 All models (both feedforward and ConvRNN) used the standard ResNet preprocessing provided 630  
631 by TensorFlow here: <https://github.com/tensorflow/tpu/blob/master/models> 631

632 `official/resnet/resnet_preprocessing.py`. Furthermore, they were trained on 632  
633 224 pixel ImageNet with stochastic gradient descent with momentum (SGDM)<sup>39</sup>, using a mo- 633  
634 mentum value of 0.9. 634

635 We allowed the base learning rate, batch size, and L2 regularization strength to vary for each 635  
636 model, depending on what was optimal in terms of top-1 validation accuracy for that model. All 636  
637 models (except for AlexNet) used the ResNet training schedule<sup>40</sup>, whereby the base learning 637  
638 rate is decayed by 90% at 30, 60, and 80 epochs, training for 90 epochs total. The AlexNet had 638  
639 its base learning rate of 0.01 subsequently decayed to 0.005, 0.001, and 0.0005, at 30, 60, and 639  
640 80 epochs, respectively. We list these values for each model in the table below: 640

641 641

<b>Model Class</b>	<b>Base Learning Rate</b>	<b>Batch Size</b>	<b>L2 Regularization</b>	
AlexNet	0.01	1024	$5 \times 10^{-4}$	
6-layer BaseNet	0.01	256	$1 \times 10^{-4}$	
642 Shallow ConvRNNs	0.01	256	$1 \times 10^{-4}$	642
11-layer BaseNet	0.0025	64	$1 \times 10^{-4}$	
ResNets	0.025	64	$1 \times 10^{-4}$	
643 Deeper ConvRNNs	0.0025	64	$1 \times 10^{-4}$	643

644 The only exceptions to the above are the models that are the result of the large-scale hyper- 644  
645 parameter searches, detailed in Section A.4. Here the learning rate and batch size are allowed 645  
646 to vary, and the L2 regularization is not uniform across the model, but is also allowed to vary 646  
647 for both the feedforward backbone and each layer’s ConvRNN cell. We list the learning rates 647  
648 and batch sizes for these models below: 648

649 649

<b>Model</b>	<b>Base Learning Rate</b>	<b>Batch Size</b>	
Shallow LSTM (“LSTM Opt” in Figure 2b)	$7.587 \times 10^{-3}$	64	
650 RGC Random	$5.184 \times 10^{-3}$	64	650
651 RGC Median	$6.736 \times 10^{-3}$	64	651

652 Since these model hyperparameters are non-standard, we manually drop the learning rate 652  
653 (using the same decay factor of 90%) once the top-1 validation accuracy saturates at that given 653

654 learning rate. 654

## 655 **A.2 Feedforward model architectures** 655

### 656 **A.2.1 BaseNet architectures** 656

657 Here we provide the architectures of the feedforward CNNs we developed in this paper, re- 657  
658 ferred to as “BaseNet” when they are later implanted with ConvRNN cells. For all of these 658  
659 architectures, we use ELU nonlinearities<sup>41</sup>. 659

660 The 6-layer BaseNet (into which we implanted ConvRNN cells to form the orange “Shallow 660  
661 ConvRNN” model class in Figure 4d), referenced as “FF” in Figure 2b, referred to as “BaseNet” 661  
662 among the blue “Shallow Feedforward” models in Figure 4d, and “Feedforward” in Figure S2c, 662  
663 had the following architecture: 663

664 664

Layer	Kernel Size	Channels	Stride	Max Pooling
1	$7 \times 7$	64	2	$2 \times 2$
2	$3 \times 3$	128	1	$2 \times 2$
3	$3 \times 3$	256	1	$2 \times 2$
4	$3 \times 3$	256	1	$2 \times 2$
5	$3 \times 3$	512	1	$2 \times 2$
6	$2 \times 2$	1000	1	No

665 665

666 666

667 The 11-layer BaseNet used for the “Deeper ConvRNNs” (green models in Figure 4d) and 667  
668 modeled after ResNet-18<sup>40</sup> (but using MaxPooling rather than stride-2 convolutions to perform 668  
669 downsampling) is given below: 669

670 670

<b>Block</b>	<b>Kernel Size</b>	<b>Depth</b>	<b>Stride</b>	<b>Max Pooling</b>	<b>Repeat</b>
1	$7 \times 7$	64	2	$2 \times 2$	$\times 1$
2	$3 \times 3$	64	1	None	$\times 2$
3	$3 \times 3$	64	1	None	$\times 2$
4	$3 \times 3$	128	1	$2 \times 2$	$\times 2$
5	$3 \times 3$	128	1	None	$\times 2$
6	$3 \times 3$	256	1	$2 \times 2$	$\times 2$
7	$3 \times 3$	256	1	$2 \times 2$	$\times 2$
8	$3 \times 3$	512	1	None	$\times 2$
9	$3 \times 3$	512	1	None	$\times 2$
10	$3 \times 3$	512	1	$2 \times 2$	$\times 2$
11	None (Avg. Pool FC)	1000	None	None	$\times 1$

This is the BaseNet used in Figure 3, among the red “Deeper Feedforward” models in Figure 4d, Figure S4, and Figure S5.

The variant of the above 6-layer feedforward CNN, referenced in Figure 2b as “FF Wider” is given below:

<b>Layer</b>	<b>Kernel Size</b>	<b>Channels</b>	<b>Stride</b>	<b>Max Pooling</b>
1	$7 \times 7$	128	2	$2 \times 2$
2	$3 \times 3$	512	1	$2 \times 2$
3	$3 \times 3$	512	1	$2 \times 2$
4	$3 \times 3$	512	1	$2 \times 2$
5	$3 \times 3$	1024	1	$2 \times 2$
6	$2 \times 2$	1000	1	None

The “FF Deeper” model referenced in Figure 2b is given below:



Layer	Kernel Size	Depth	Stride	Max Pooling
1	$7 \times 7$	64	2	$2 \times 2$
2	$3 \times 3$	64	1	None
3	$3 \times 3$	64	1	None
4	$3 \times 3$	128	1	$2 \times 2$
5	$3 \times 3$	128	1	None
6	$3 \times 3$	256	1	$2 \times 2$
7	$3 \times 3$	256	1	$2 \times 2$
8	$3 \times 3$	512	1	None
9	$3 \times 3$	512	1	None
10	$3 \times 3$	512	1	$2 \times 2$
11	None (Avg. Pool FC)	1000	None	None

## A.2.2 AlexNet

We use the standard AlexNet architecture, which uses local response normalization<sup>26</sup>. We note that we are able to attain a higher than reported top-1 validation accuracy of 63.9% (compared to 57% accuracy) by using the ResNet preprocessing mentioned in Section A.1.3.

## A.2.3 ResNet Architectures

For the ResNet architectures, we used the original v1 versions<sup>40</sup> for ResNet-18 and ResNet-34. For deeper ResNets (ResNet-50, ResNet-101, and ResNet-152), we used the v2 variants of ResNets, as this gave them a slightly higher increase in top-1 ImageNet validation accuracy. Specifically, the v2 variants of ResNets use the pre-activation of the weight layers rather than the post-activation used in the original versions. Furthermore, the v2 variants of ResNets apply batch normalization<sup>42</sup> and ReLU to the input *prior* to the convolution, whereas the original variants apply these operations after the convolution. We use the TensorFlow Slim implementations for these two variants provided here: <https://github.com/tensorflow/models/tree/master/research/slim>.

### 698 **A.3 ConvRNN Cell Equations** 698

699 Here we provide the explicit update equations for each of the ConvRNN cells referenced in the 699  
700 barplot in Figure 2d ( $C_\ell$  in (1)), where  $\sigma$  denotes the sigmoid function. 700

701 Throughout these sections, we let  $\circ$  denote Hadamard (elementwise) product, let  $*$  denote 701  
702 convolution, let  $h_t^\ell$  denote the output of the cell, let  $s_t^\ell$  denote the propagated memory of the cell 702  
703 (also known as the hidden state), and let  $x_t^\ell = \bigoplus_{j \neq \ell} r_t^j$  denote the input to the cell at layer  $\ell$  (this 703  
704 is the concatenation of feedforward and feedback inputs to layer  $\ell$ , defined in Section A.1.2). 704

705 In the following table, we provide the number of timesteps the ConvRNNs were unrolled 705  
706 for, and the timestep at which the image presentation was replaced by a mean gray stimulus 706  
707 during model training: 707

708 <b>Model Class</b>	<b>Unroll Timesteps</b>	<b>Image Presentation Off Timestep</b>
709 Shallow ConvRNNs	16	12
710 Deeper ConvRNNs	17	12
RGC Random	26	10

711 These parameters were chosen based on what yielded high performance for that model class 711  
712 and also what was able to feasibly fit into TPU memory for training (more unroll timesteps re- 712  
713 quires more memory, but can also lead to instability during training, as is common with training 713  
714 RNNs<sup>43</sup>). 714

715 For the “Shallow ConvRNNs”, ConvRNN cells were implanted into convolutional layers 3, 715  
716 4, and 5 of the 6-layer BaseNet. For the “Deeper ConvRNNs”, ConvRNN cells were implanted 716  
717 into convolutional layers 4, 5, 6, 7, 8, 9, and 10 of the 11-layer BaseNet. 717

### 718 **A.3.1 Time Decay** 718

719 This is the simplest form of recurrence that we consider and has a discrete-time trajectory given 719  
720 by 720

$$\begin{aligned} s_t^\ell &= F_\ell(x_t^\ell) + \tau_\ell s_{t-1}^\ell \\ h_t^\ell &= s_t^\ell, \end{aligned} \quad (2)$$

721 where  $\tau_\ell$  is the learned time constant at a given layer  $\ell$ . This model is intended to be a control 721  
722 for simplicity, where the time constants could model synaptic facilitation and depression in a 722  
723 cortical layer. 723

724 For the TensorFlow implementation of this cell, see the `GenFuncCell()` class and its 724  
725 associated `memory()` function in the `cell.py` file of our TNN Github repository. 725

### 726 **A.3.2 SimpleRNN** 726

727 The update equations in this case are given by: 727

$$\begin{aligned} a_t^\ell &= W_s^\ell * s_{t-1}^\ell + b_s^\ell \\ i_t^\ell &= W_i^\ell * x_t^\ell + b_i^\ell \\ s_t^\ell &= \text{elu}(\text{LN}(i_t^\ell + a_t^\ell)) \\ h_t^\ell &= s_t^\ell, \end{aligned} \quad (3)$$

728 where LN denotes the layer normalization operation<sup>44</sup> with offset parameter  $\beta$  initialized to 0 728  
729 and scale parameter  $\gamma$  initialized to 1. For the shallow SimpleRNN (among the orange “Shallow 729  
730 ConvRNN” models in Figure 4d), we use layer normalization but omit its usage in the deeper 730  
731 ConvRNN as it was not able to train with that operation. 731

732 For the TensorFlow implementation of this cell, see the `ConvNormBasicCell()` class 732  
733 in the `convrnn.py` file of our TNN Github repository. 733

### 734 A.3.3 GRU 734

735 We adapt the standard GRU cell<sup>45</sup> to the convolutional setting: 735

$$\begin{aligned}
 r_t^\ell &= \sigma(W_r^\ell * x_t^\ell + U_r^\ell * s_{t-1}^\ell + b_r^\ell + 1) \\
 u_t^\ell &= \sigma(W_u^\ell * x_t^\ell + U_u^\ell * s_{t-1}^\ell + b_u^\ell) \\
 c_t^\ell &= \tanh(W_c^\ell * x_t^\ell + U_c^\ell * (r_t^\ell \circ s_{t-1}^\ell) + b_c^\ell) \\
 s_t^\ell &= u_t^\ell \circ s_{t-1}^\ell + (1 - u_t^\ell) \circ c_t^\ell \\
 h_t^\ell &= s_t^\ell.
 \end{aligned} \tag{4}$$

736 For the TensorFlow implementation of this cell, see the `ConvGRUCell()` class in the `convrnn.py` file of our TNN Github repository. 736  
 737 737

### 738 A.3.4 LSTM 738

739 We adapt the standard LSTM cell<sup>46</sup> to the convolutional setting, with some slight modifications 739  
 740 such as added layer normalization for stability in training. 740

741 We first make the gates convolutional as follows: 741

$$\begin{aligned}
 i_t^\ell &= LN(W_i^\ell * x_t^\ell + U_i^\ell * h_{t-1}^\ell + b_i^\ell) \\
 j_t^\ell &= LN(W_j^\ell * x_t^\ell + U_j^\ell * h_{t-1}^\ell + b_j^\ell) \\
 f_t^\ell &= LN(W_f^\ell * x_t^\ell + U_f^\ell * h_{t-1}^\ell + b_f^\ell) \\
 o_t^\ell &= LN(W_o^\ell * x_t^\ell + U_o^\ell * h_{t-1}^\ell + b_o^\ell),
 \end{aligned} \tag{5}$$

742 where LN denotes the layer normalization operation<sup>44</sup> with offset parameter  $\beta$  initialized to 0 742  
 743 and scale parameter  $\gamma$  initialized to 1. 743

744 Next, the LSTM update equations are as follows: 744

$$\begin{aligned}
 s_t^\ell &= LN(s_{t-1}^\ell \circ \sigma(f_t^\ell + f_b^\ell) + \sigma(i_t^\ell) \circ \tanh(j_t^\ell)) \\
 h_t^\ell &= \tanh(s_t^\ell) \circ \sigma(o_t^\ell),
 \end{aligned} \tag{6}$$

745 where  $f_b^\ell$  is the forget gate bias, typically set to 1, as recommended by others<sup>47</sup>. When peephole  
746 connections<sup>48</sup> are allowed, these update equations are augmented to become:

$$\begin{aligned} s_t^\ell &= LN(s_{t-1}^\ell \circ \sigma(f_t^\ell + f_b^\ell + V_f^\ell \circ s_{t-1}^\ell) + \sigma(i_t^\ell + V_i^\ell \circ s_{t-1}^\ell) \circ \tanh(j_t^\ell)) \\ h_t^\ell &= \tanh(s_t^\ell) \circ \sigma(o_t^\ell + V_o^\ell \circ s_{t-1}^\ell). \end{aligned} \quad (7)$$

747 In the shallow LSTM (among the orange “Shallow ConvRNN” models in Figure 4d), we use  
748 peepholes and layer normalization, as that was found in the LSTM search for shallow models  
749 (described in Section A.4.1) to be useful for performance. We found, however, that neither of  
750 these augmentations are needed in the deeper variant (among the green “Deeper ConvRNN”  
751 models in Figure 4d) in order to achieve high top-1 validation accuracy on ImageNet.

752 For the TensorFlow implementation of this cell, see the `ConvLSTMCell()` class in the  
753 `convrnn.py` file of our TNN Github repository.

### 754 A.3.5 UGRNN 754

755 We adapt the UGRNN<sup>25</sup> to the convolutional setting. The update equations are as follows:

$$\begin{aligned} c_t^\ell &= \tanh(W_c^\ell * x_t^\ell + U_c^\ell * s_{t-1}^\ell + b_c^\ell) \\ g_t^\ell &= \sigma(W_g^\ell * x_t^\ell + U_g^\ell * s_{t-1}^\ell + b_g^\ell + 1) \\ s_t^\ell &= g_t^\ell \circ s_{t-1}^\ell + (1 - g_t^\ell) \circ c_t^\ell \\ h_t^\ell &= s_t^\ell. \end{aligned} \quad (8)$$

756 For the TensorFlow implementation of this cell, see the `ConvUGRNCell()` class in the  
757 `convrnn.py` file of our TNN Github repository.

### 758 **A.3.6 IntersectionRNN** 758

759 We adapt the IntersectionRNN<sup>25</sup> to the convolutional setting. The update equations are as 759  
 760 follows: 760

$$\begin{aligned}
 m_t^\ell &= \tanh(W_m^\ell * x_t^\ell + U_m^\ell * s_{t-1}^\ell + b_m^\ell) \\
 n_t^\ell &= \text{relu}(W_n^\ell * x_t^\ell + U_n^\ell * s_{t-1}^\ell + b_n^\ell) \\
 p_t^\ell &= \sigma(W_p^\ell * x_t^\ell + U_p^\ell * s_{t-1}^\ell + b_p^\ell + 1) \\
 y_t^\ell &= \sigma(W_y^\ell * x_t^\ell + U_y^\ell * s_{t-1}^\ell + b_y^\ell + 1) \\
 s_t^\ell &= p_t^\ell \circ s_{t-1}^\ell + (1 - p_t^\ell) \circ m_t^\ell \\
 h_t^\ell &= y_t^\ell \circ x_t^\ell + (1 - y_t^\ell) \circ n_t^\ell.
 \end{aligned} \tag{9}$$

761 For the TensorFlow implementation of this cell, see the `ConvIntersectionRNNCell()` 761  
 762 class in the `convrnn.py` file of our TNN Github repository. 762

### 763 **A.3.7 Reciprocal Gated Cell (RGC)** 763

764 Here we provide the explicit update equations for the Reciprocal Gated Cell<sup>27</sup>, diagrammed in 764  
 765 Figure 2a (bottom right). The update equation for the output of the cell,  $h_t^\ell$ , is given by a gating 765  
 766 of both the input and memory  $s_t^\ell$ : 766

$$\begin{aligned}
 a_{t+1}^\ell &= (1 - \sigma(W_{sh}^\ell * s_t^\ell)) \circ x_t^\ell + (1 - \sigma(W_{hh}^\ell * h_t^\ell)) \circ h_t^\ell \\
 h_t^\ell &= \text{elu}(a_t^\ell).
 \end{aligned} \tag{10}$$

767 The update equation for the memory  $s_t^\ell$  is given by a gating of the input and the output of 767  
 768 the cell  $h_t^\ell$ : 768

$$\begin{aligned}
 \tilde{s}_{t+1}^\ell &= (1 - \sigma(W_{hs}^\ell * h_t^\ell)) \circ x_t^\ell + (1 - \sigma(W_{ss}^\ell * s_t^\ell)) \circ s_t^\ell \\
 s_t^\ell &= \text{elu}(\tilde{s}_t^\ell).
 \end{aligned} \tag{11}$$

769 For the TensorFlow implementation of this cell, see the `ReciprocalGateCell()` class 769  
 770 in the `reciprocalgaternn.py` file of our TNN Github repository. 770

## 771 **A.4 ConvRNN Searches** 771

772 We employed a form of Bayesian optimization, a Tree-structured Parzen Estimator (TPE), 772  
773 to search the space of continuous and categorical hyperparameters<sup>49</sup>. This algorithm con- 773  
774 structs a generative model of  $P[\text{score} \mid \text{configuration}]$  by updating a prior from a maintained 774  
775 history  $H$  of hyperparameter configuration-loss pairs. The fitness function that is optimized 775  
776 over models is the expected improvement, where a given configuration  $c$  is meant to optimize 776  
777  $EI(c) = \int_{x < t} P[x \mid c, H]$ . This choice of Bayesian optimization algorithm models  $P[c \mid x]$  via 777  
778 a Gaussian mixture, and restricts us to tree-structured configuration spaces. 778

779 Models were trained synchronously 100 models at a time using the HyperOpt package<sup>50</sup>, 779  
780 which implements the above Bayesian optimization. Each model was trained on its own Tensor 780  
781 Processing Unit (TPUv2), and during the search, ConvRNN models were trained by stochastic 781  
782 gradient descent on 128 pixel ImageNet for efficiency. The top performing ConvRNN models 782  
783 were then fully trained out on 224 pixel ImageNet. 783

### 784 **A.4.1 LSTM search** 784

785 The search for better LSTM architectures involved searching over training hyperparameters 785  
786 and common structural variants of the LSTM to better adapt this local structure to deep con- 786  
787 volutional networks, using hundreds of second generation Google Tensor Processing Units 787  
788 (TPUv2s). We searched over learning hyperparameters (e.g. gradient clip values, learning rate) 788  
789 as well as structural hyperparameters (e.g. gate convolution filter sizes, channel depth, whether 789  
790 or not to use peephole connections, etc.). 790

791 Specifically, we implanted LSTMs into convolutional layers 3, 4, and 5, of the 6-layer 791  
792 BaseNet described in Section A.2. At each of these layers, the parameters of the LSTM cell 792  
793 (defined in Section A.3.4) were allowed to vary per layer, as follows: 793

- 794 • The discrete number of convolutional channels was chosen from  $\{64, 128, 256\}$ . 794

- 795 • The discrete choice of convolutional filter sizes were chosen from  $\{1, 4\}$ . 795
  - 796 • The binary choice of whether or not to use layer normalization. 796
  - 797 • The strength of the L2 regularization of all LSTM parameters in that layer  $\in [10^{-7}, 10^{-3}]$ , 797  
798 sampled log-uniformly. 798
  - 799 • The scale of the He-style initialization<sup>51</sup> of the convolutional filter weights  $\in [0.25, 2]$ , 799  
800 sampled uniformly. 800
  - 801 • The value of the constant initialization of the biases  $\in [-2, 2]$ , sampled uniformly. 801
  - 802 • The forget gate bias  $f_b^\ell \in [0, 6]$ , sampled uniformly (defined in (6)). 802
  - 803 • The binary choice of whether or not to use peephole connections (as defined in (7)). 803
- 804 Outside of the LSTM cell at each layer, we additionally searched over the following param- 804  
805 eters as well: 805
- 806 • The number of discrete timesteps the model is unrolled  $\in [12, 26]$ , sampled uniformly in 806  
807 consecutive groups of size 2. 807
  - 808 • The timestep at each the image presentation is “turned off” and replaced with a mean 808  
809 gray stimulus  $\in [8, 12]$ , sampled uniformly in consecutive groups of 2. 809
  - 810 • The discrete choice of batch size used for the training the entire model  $\in \{64, 128, 256\}$ . 810
  - 811 • The learning rate for training the entire model  $\in [10^{-3}, 10^{-1}]$ , sampled log-uniformly. 811
  - 812 • The binary choice of whether or not to use Nesterov momentum<sup>52</sup>. 812
  - 813 • The gradient clipping value  $\in [0.3, 3]$ , sampled log-uniformly. 813



- 814 • The scale of the He-style initialization<sup>51</sup> of the convolutional filter weights of the feed- 814  
815 forward base model  $\in [0.25, 2]$ , sampled uniformly. 815
- 816 • The strength of the L2 regularization of the feedforward base model parameters  $\in [10^{-7}, 10^{-3}]$ , 816  
817 sampled log-uniformly. 817

818 Each search point is a sampled value from the above described search space and trained for 1 818  
819 epoch on ImageNet, in order to sample as many models as much as possible with the compu- 819  
820 tational resources available. More than 1600 models were sampled in total, and we trained out 820  
821 the top ones and the median performing one after 1 epoch were trained out fully on 224 pixel 821  
822 ImageNet. The median model from this search attained the best top-1 validation accuracy on 822  
823 ImageNet, which is the resultant “LSTM Opt” model in Figure 2b. 823

#### 824 **A.4.2 Reciprocal Gate Cell (RGC) search** 824

825 From the Reciprocal Gated Cell equations in (10) and (11), there are a variety of possibilities 825  
826 for how  $h_{t-1}^l$ ,  $x_t^l$ ,  $s_t^l$ , and  $h_t^l$  can be connected to one another (schematized in Figure 2c). 826

Mathematically, the search in Figure 2c can be formalized in terms of the following update equations. First, we define our input sets and building block functions:

$$minin = \{h_{t-1}^{l-1}, x_t^l, s_{t-1}^l, h_{t-1}^l\}$$

$$minin_a = minin \cup \{s_t^l\}$$

$$minin_b = minin \cup \{h_t^l\}$$

$$S_a \subseteq minin_a$$

$$S_b \subseteq minin_b$$

$$\text{Affine}(x) \in \{+, 1 \times 1 \text{ conv}, K \times K \text{ conv}, K \times K \text{ depth-separable conv}\}$$

$$K \in \{3, \dots, 7\}$$

With those in hand, we have the following update equations:

$$\begin{aligned}\tau_a &= v_1^\tau + v_2^\tau \sigma(\text{Affine}(S_a)) \\ \tau_b &= v_1^\tau + v_2^\tau \sigma(\text{Affine}(S_b)) \\ gate_a &= v_1^g + v_2^g \sigma(\text{Affine}(S_a)) \\ gate_b &= v_1^g + v_2^g \sigma(\text{Affine}(S_b)) \\ a_t^\ell &= \{gate_a\} \cdot in_t^\ell + \{\tau_a\} \cdot h_{t-1}^\ell \\ h_t^\ell &= f(a_t^\ell) \\ b_t^\ell &= \{gate_b\} \cdot in_t^\ell + \{\tau_b\} \cdot s_{t-1}^\ell \\ s_t^\ell &= f(b_t^\ell) \\ f &\in \{\text{elu}, \text{tanh}, \sigma\}.\end{aligned}$$

827 For clarity, the following matrix summarizes the connectivity possibilities (with ? denoting the  
828 possibility of a connection), schematized in Figure 2c: 828

$$\begin{matrix} & h_{t-1}^{\ell-1} & x_t^\ell & s_{t-1}^\ell & s_t^\ell & h_{t-1}^\ell & h_t^\ell \\ \begin{matrix} h_{t-1}^{\ell-1} \\ x_t^\ell \\ s_{t-1}^\ell \\ s_t^\ell \\ h_{t-1}^\ell \\ h_t^\ell \end{matrix} & \left( \begin{array}{cccccc} 0 & 1 & 0 & ? & 0 & ? \\ 0 & 0 & 0 & ? & 0 & ? \\ 0 & 0 & 0 & ? & 0 & ? \\ 0 & 0 & 0 & 0 & 0 & ? \\ 0 & 0 & 0 & ? & 0 & ? \\ 0 & 0 & 0 & ? & 0 & 0 \end{array} \right)\end{matrix}$$

829 Each search point is a sampled value from the above described search space and trained 829  
830 for 5 epochs on ImageNet, in order to sample as many models as much as possible with the 830  
831 computational resources available. Around 6000 models were sampled in total over the course 831  
832 of the search. The top and median models from this search were then fully trained out on 832  
833 224 pixel ImageNet with a batch size of 64 (which was maximum that we could fit into TPU 833  
834 memory). Moreover, as explicated in the table in Section A.1.3, the ResNet models were also 834  
835 trained using this same batch size, with the standard ResNet learning rate of 0.1 for a batch 835  
836 size of 256 linearly rescaled to accomodate, to ensure fair comparison between these two model 836

837 classes. The median model from this search attained the best top-1 validation accuracy on 837  
838 ImageNet of all models selected to be trained out fully on ImageNet from the search, producing 838  
839 the resultant “RGC Median” model in Figure 2d (note that this designation also includes the 839  
840 long-range feedback connections). The “RGC Random” model is from the random phase of 840  
841 this search (400th sampled model, since models sampled earlier than that failed to train out 841  
842 fully on ImageNet). 842

## 843 **A.5 Decoders** 843

844 In addition to choice of ConvRNN cell, we consider particular choices of “light-weight” (in 844  
845 terms of parameter count) decoding strategy that determines the final object category of that 845  
846 image. By construction, the model will output category logit probabilities at each timestep, 846  
847 given by the softmax function  $\text{softmax}(z; \beta) = \frac{e^{\beta z_i}}{\sum_{j=1}^C e^{\beta z_j}}$ , where  $C = 1000$  is the number of 847  
848 ImageNet categories. This will then be passed to a decoding function which can take one of 848  
849 several forms: 849

- 850 1. **Default:** Use the logits at the last timestep and discard the remaining, with  $\beta = 1$ . 850
- 851 2. **Threshold Decoder:** Select the logits from the first timepoint at which the maximum 851  
852 logit value at that timepoint crosses a fixed threshold (set to 0.9), with  $\beta = 1$ . 852
- 853 3. **Max Confidence Decoder:** For the most confident category, find the timepoint at which 853  
854 that confidence peaks, and return the logits at that timepoint, where  $\beta$  is a trainable scalar 854  
855 parameter initialized to 1. 855

856 “RGC Median” therefore refers to the model trained using the default decoder, but when using 856  
857 the other two decoders with the “RGC Median” model, we append it to the name (as is done in 857  
858 Figures 4d, S4a, and S5). 858

## 859 **A.6 Model prediction of neural responses** 859

### 860 **A.6.1 Neural data** 860

861 Neural responses came from three multi-unit arrays per primate (rhesus macques): one im- 861  
862 planted in V4, one in posterior IT (pIT), and one in central and anterior IT (cIT/aIT)<sup>29</sup>. Each 862  
863 image was presented approximately 50 times, using rapid visual stimulus presentation (RSVP). 863  
864 Each stimulus was presented for 100ms, followed by a mean gray stimulus interleaved between 864  
865 images. Each trial lasted 260ms. The image set consisted of 5120 images based on 64 ob- 865  
866 ject categories. These objects belonged to 8 high-level categories (tables, planes, fruits, faces, 866  
867 chairs, cars, boats, animals), each of which consisted of 8 unique objects. Each image consisted 867  
868 of a 2D projection of a 3D model added to a random background. The pose, size, and  $x$ - and 868  
869  $y$ -position of the object was varied across the image set, whereby 2 levels of variation were 869  
870 used (corresponding to medium and high variation<sup>29</sup>). Multi-unit responses to these images 870  
871 were binned in 10ms windows, averaged across trials of the same image, and normalized to 871  
872 the average response to a blank image. This produced a set of 5120 images x 256 units x 25 872  
873 timebins responses, which were the targets for our model features to predict. There were 88 873  
874 units from V4, 88 units from pIT, and 80 units from cIT/aIT. 874

### 875 **A.6.2 Fitting procedure** 875

876 *Generating train/test split.* The 5120 images were split 75%-25% within each object category 876  
877 into a training set and a held-out testing set. All images were presented to the models for 10 877  
878 time steps (corresponding to 100ms), followed by a mean gray stimulus for the remaining 15 878  
879 time steps, to match the image presentation to the primates. The images are matched to the 879  
880 procedure when used to validate the models on ImageNet, namely they are bilinearly resized to 880  
881  $224 \times 224$  and normalized by the ImageNet mean ( $[0.485, 0.456, 0.406]$ ) and standard deviation 881  
882 ( $[0.229, 0.224, 0.225]$ ), applied per channel. 882

883 *Model layer determination.* We stipulated that units from each multi-unit array must be 883  
884 fit by features from a single model layer. To determine which one, we fit the features from 884  
885 the relevant feedforward BaseNet (either the 6-layer BaseNet or 11-layer BaseNet) to unit's 885  
886 time-averaged response, and counted how many units had minimal loss for a given model layer, 886  
887 schematized in Step 2 of Figure 1. This yielded a mapping from the V4 array to model layer 3 887  
888 of the 6-layer BaseNet and model layers 5 & 6 of the 11-layer BaseNet, pIT mapping to model 888  
889 layer 4 of the 6-layer BaseNet and model layers 7 & 8 of the 11-layer BaseNet, and cIT/aIT 889  
890 mapping to layer 5 of the 6-layer BaseNet and model layers 9 & 10 of the 11-layer BaseNet. 890

891 *Mapping transform from models to neural responses.* Model features from each image 891  
892 (i.e. the activations of units in a given model layer) were linearly fit to the neural responses 892  
893 by stochastic gradient descent with a standard L2 loss using a spatially factored mapping<sup>53</sup>, 893  
894 where each of the 256 units was fit independently. This spatially factored mapping is defined as 894  
895 follows: Given a model feature  $f^\ell \in \mathbb{R}^{x,y,c}$  from layer  $\ell$ , where  $x$  and  $y$  are the number of units 895  
896 in the spatial extent and  $c$  is the number of channels, we fit a spatial mask  $w_{\text{space}} \in \mathbb{R}^{x,y}$  and 896  
897 a channel mask  $w_{\text{channels}} \in \mathbb{R}^c$  for each neuron  $n$  to predict the ground-truth neuron's response 897  
898  $r_{i,n,t}$  at image  $i$  and timebin  $t$ . The predicted response can be written as: 898

$$\hat{r}_{i,n,t;w} = \sum_{i=1}^x \sum_{j=1}^y \sum_{k=1}^c w_{\text{space}}[i,j] w_{\text{channels}}[k] f^\ell[i,j,k]. \quad (12)$$

899 This mapping is implemented in the `factored_fc()` function of the `cell.py` file of our 899  
900 TNN Github repository. 900

901 *Loss function.* After these layers were determined, model features were then fit to the entire 901  
902 set of 25 timebins for each unit using a shared linear model: that is, a single set of regression 902  
903 coefficients was used for all timebins, as schematized in Step 3 of Figure 1. The loss for this 903  
904 fitting was the average L2 loss across training images and 25 timebins for each unit, given by 904

$$\mathcal{L}(\hat{r}_{i,n,t;w}, r_{i,n,t}) = \frac{1}{|\mathcal{B}|} \sum_{t=6}^{25} \sum_{i \in \mathcal{B}} \sum_{n=1}^{256} (\hat{r}_{i,n,t;w} - r_{i,n,t})^2. \quad (13)$$

905 Note that  $t$  indexes model timesteps, which correspond to  $10ms$  timebins, so  $t = 6$  refers to the 905  
906  $60-70ms$  timebin,  $t = 7$  refers to the  $70-80ms$  timebin, and so forth. 906

907 We trained the temporally-fixed parameters  $w = [w_{\text{space}}; w_{\text{channels}}]$  of the mapping using the 907  
908 Adam optimizer<sup>54</sup> with a learning rate of  $1 \times 10^{-4}$  and a training batch size  $|\mathcal{B}| = 64$  images. 908  
909 Additionally, we used a dropout<sup>55</sup> level of 0.5 on the model features, prior to the mapping, as 909  
910 further regularization. 910

### 911 **A.6.3 Metrics** 911

912 To estimate a noise ceiling for each neuron’s response at each timebin, we computed the 912  
913 Spearman-Brown corrected split-half reliability  $\rho_n$  of neuron  $n$ , averaged across 900 bootstrap 913  
914 iterations of split-half trials. 914

915 Let “Neural Predictivity” (used in Figure S2) refer to 915

$$\text{Corr}(\hat{r}^{\text{test}}, r_n^{\text{test}}), \quad (14)$$

916 namely the Pearson correlation across test set images of the model’s response  $\hat{r}^{\text{test}}$  to the of any 916  
917 neuron  $n$ ’s response  $r_n^{\text{test}}$  at a given timebin (or time-averaged). 917

918 The “Neural Predictivity (Noise Corrected)” (used in Figure 3 and Figure S4) for neuron  $n$  918  
919 is given by 919

$$\frac{\text{Corr}(\hat{r}^{\text{test}}, r_n^{\text{test}})}{\sqrt{\rho_n}}. \quad (15)$$

## 920 **A.7 Inter-animal consistency** 920

921 We provide the definition and justification of the inter-animal consistency metric mentioned in 921  
922 Figure S4b. Suppose we have neural responses from two primates  $A$  and  $B$ . Let  $t_i^p$  be the vector 922  
923 of true responses (either at a given timebin or averaged across a set of timebins) of primate 923  
924  $p \in \{A, B\}$  on stimulus set  $i \in \{\text{train}, \text{test}\}$ . Of course, we only receive noisy observations of 924  
925  $t_i^p$ , so let  $s_{j,i}^p$  be the  $j$ -th set of  $n$  trials of  $t_i^p$ . Finally, let  $M(x)_i$  be the predictions of a mapping 925

926  $M$  (e.g. PLS) when trained on input  $x$  and tested on stimulus set  $i$ . For example,  $M(t_{\text{train}}^p)_{\text{test}}$  926  
 927 is the prediction of the mapping  $M$  on the test stimulus trained on the true neural responses 927  
 928 from primate  $p$  on the train stimulus, and correspondingly,  $M(s_{1,\text{train}}^p)_{\text{test}}$  is the prediction of the 928  
 929 mapping  $M$  on the test stimulus trained on the (trial-average) of noisy sample 1 on the train 929  
 930 stimulus from primate  $p$ . 930

931 With these definitions in hand, the inter-animal mapping consistency from one primate  $A$  to 931  
 932 another primate  $B$  corresponds to the following true quantity to be estimated: 932

$$\text{Corr}(M(t_{\text{train}}^A)_{\text{test}}, t_{\text{test}}^B), \quad (16)$$

933 where Corr is the Pearson correlation across test stimuli. In what follows, we argue that this 933  
 934 true quantity can be approximated with the following ratio of measurable quantities where we 934  
 935 divide the noisy trial observations into two sets of equal samples: 935

$$\text{Corr}(M(t_{\text{train}}^A)_{\text{test}}, t_{\text{test}}^B) \sim \frac{\text{Corr}(M(s_{1,\text{train}}^A)_{\text{test}}, s_{2,\text{test}}^B)}{\sqrt{\text{Corr}(M(s_{1,\text{train}}^A)_{\text{test}}, M(s_{2,\text{train}}^A)_{\text{test}}) \times \text{Corr}(s_{1,\text{test}}^B, s_{2,\text{test}}^B)}}. \quad (17)$$

936 In words, the inter-animal consistency corresponds to the predictivity of the mapping on the test 936  
 937 set stimuli from primate  $A$  to  $B$  on two different (averaged) halves of noisy trials, corrected by 937  
 938 the square root of the mapping reliability on primate  $A$ 's test stimuli responses on two different 938  
 939 halves of noisy trials and the internal consistency of primate  $B$ . 939

940 We justify the approximation in (17) by gradually eliminating the true quantities by their 940  
 941 measurable estimates, starting from the original quantity in (16). First, we make the approxi- 941  
 942 mation that 942

$$\text{Corr}(M(t_{\text{train}}^A)_{\text{test}}, s_{2,\text{test}}^B) \sim \text{Corr}(M(t_{\text{train}}^A)_{\text{test}}, t_{\text{test}}^B) \times \text{Corr}(t_{\text{test}}^B, s_{2,\text{test}}^B). \quad (18)$$

943 by transitivity of positive correlations (which is reasonable assumption when the number of 943  
 944 stimuli is large). Next, by normality assumptions in the structure of the noisy estimates and 944

945 since the number of trials ( $n$ ) between the two sets is the same, we have that

945

$$\text{Corr} (s_{1,\text{test}}^B, s_{2,\text{test}}^B) \sim \text{Corr} (t_2^B, s_{2,\text{test}}^B)^2. \quad (19)$$

946 Namely, the correlation between the average of two sets of noisy observations of  $n$  trials each

946

947 is approximately the square of the correlation between the true value and average of one set of

947

948  $n$  noisy trials. Therefore, from (18) and (19) it follows that

948

$$\text{Corr} (M (t_{\text{train}}^A)_{\text{test}}, t_{\text{test}}^B) \sim \frac{\text{Corr} (M (t_{\text{train}}^A)_{\text{test}}, s_{2,\text{test}}^B)}{\sqrt{\text{Corr} (s_{1,\text{test}}^B, s_{2,\text{test}}^B)}}. \quad (20)$$

We have gotten rid of  $t_2^B$ , but we still need to get rid of the  $M (t_{\text{train}}^A)_{\text{test}}$  term. We apply the same two steps by analogy though these approximations may not always be true (though are true for additive Gaussian noise):

$$\begin{aligned} \text{Corr} (M (s_{1,\text{train}}^A)_{\text{test}}, s_{2,\text{test}}^B) &\sim \text{Corr} (s_{2,\text{test}}^B, M (t_{\text{train}}^A)_{\text{test}}) \times \text{Corr} (M (t_{\text{train}}^A)_{\text{test}}, M (s_{1,\text{train}}^A)_{\text{test}}) \\ \text{Corr} (M (s_{1,\text{train}}^A)_{\text{test}}, M (s_{2,\text{train}}^A)_{\text{test}}) &\sim \text{Corr} (M (s_{1,\text{train}}^A)_{\text{test}}, M (t_{\text{train}}^A)_{\text{test}})^2, \end{aligned}$$

949 which taken together implies

949

$$\text{Corr} (M (t_{\text{train}}^A)_{\text{test}}, s_{2,\text{test}}^B) \sim \frac{\text{Corr} (M (s_{1,\text{train}}^A)_{\text{test}}, s_{2,\text{test}}^B)}{\sqrt{\text{Corr} (M (s_{1,\text{train}}^A)_{\text{test}}, M (s_{2,\text{train}}^A)_{\text{test}})}}. \quad (21)$$

950 Equations (20) and (21) together imply the final estimated quantity given in (17).

950

## 951 **A.8 Object solution times (OSTs)**

951

### 952 **A.8.1 Generating model OSTs**

952

953 Here we describe how we defined object solution times from both feedforward models and

953

954 ConvRNNs. As depicted in Figure 4a, this is a multi-stage process that involves first identifying

954

955 the most “IT-preferred” layers of each model.

955



956 *Determining “IT-preferred” model layers.* These are identified by a standard<sup>4,17</sup> linear map- 956  
957 ping using 25 component partial least squares regression (PLS), from model layer units to *time-* 957  
958 *averaged* IT (namely, pIT/cIT/aIT) responses from the neural data described in Section A.6.1, 958  
959 and corroborates the results obtained by the same procedure described in Section A.6.2. We 959  
960 use this neural data as it has both V4 and IT responses, and demonstrates a disjoint set of layers 960  
961 between the preferred V4 model layers and preferred IT layers. 961

962 *Mapping model timepoints to IT timepoints.* Once these “IT-preferred” model layers are 962  
963 identified, we then map these model timepoints to 10ms timebins as in the IT data. For Con- 963  
964 vRNNs with intrinsic temporal dynamics, this mapping is one-to-one, we simply concatenate 964  
965 the model layers at each timepoint to construct an entire IT pseudopopulation, and each time- 965  
966 point of the ConvRNN corresponds to a 10ms timebin between 70-260ms. For feedforward 966  
967 models, we map each “IT-preferred” layer to a 10ms timebin between 70-260ms. If the number 967  
968 of “IT-preferred” layers for a feedforward model matches the total number of timebins (19), 968  
969 then there is only one admissible mapping, corresponding to the “uniform” mapping, whereby 969  
970 the earliest (in the feedforward hierarchy) layer is matched to the earliest 10ms timebin of 70ms, 970  
971 and so forth. On the other hand, if the number of “IT-preferred” layers is strictly less than the 971  
972 total number of timebins, then we additionally consider a “graded” mapping that picks a ran- 972  
973 dom sample of units from one layer to the next so that the number of feedforward layers exactly 973  
974 matches the total number of timebins. 974

975 *Obtaining model  $d'$  values.* Once a timepoint mapping is selected, we compute the model 975  
976 object solution time (OST) in the same manner as the OST is computed for IT<sup>17</sup>. Specifically, 976  
977 we train an SVM ( $C = 5 \times 10^4$ ) separately for each model timepoint after it has been dimension 977  
978 reduced through PCA (with 1000 components) to solve the ten-way categorization task for each 978  
979 image. The ten categories are apple, bear, bird, car, chair, dog, elephant, person, plane, and 979  
980 zebra. 1000 images constitute the training set of the SVM (100 images per category) and 320 980

981 images are randomly chosen to be in the test set. We perform 20 trials each of 10 train/test 981  
982 splits to get errorbars, where each image is in the test set at least once. The (model or IT)  $d'$  982  
983 for that image is only computed from the SVM when it has been in the test set, and is bounded 983  
984 between -5 and 5. Since this dataset consists of 1,320 grayscale images presented centrally to 984  
985 behaving primates for 100ms, there are therefore 1,320  $d'$  values (one for each image) for any 985  
986 given model, constituting its “I1” vector<sup>30</sup>. 986

987 *Correlating model OST with IT OST.* The OST of the model therefore is the first model 987  
988 timepoint in which the  $d'$  reaches the recorded primate  $d'$  for that image. Using the Leven- 988  
989 berg–Marquardt algorithm, we further linearly interpolate between 10ms bins to determine the 989  
990 precise millisecond that the response surpassed the primate’s behavioral output for that image 990  
991 (as was done analogously with the IT population’s OST). Finally, we compare the model OST 991  
992 to the IT OST via a Spearman correlation across the common set of images solved by *both* the 992  
993 model and IT. 993

#### 994 **A.8.2 Relating the linear mapping to neural responses with the OST behavioral metric** 994

995 The IT population OST was computed from primarily anterior IT (aIT) responses<sup>17</sup>. Therefore, 995  
996 to isolate the interaction a linear mapping of model features to neural responses (as we do in 996  
997 neural response prediction described in Section A.6) might have compared to directly comput- 997  
998 ing the OST from the original model features, we turned to neural data collected from 486 aIT 998  
999 units on 1100 greyscale images. 999

1000 For each model, we train a linear mapping on this dataset, with 550 images used for training 1000  
1001 the mapping and 550 images are held-out for the test set. We observe similar conclusions as with 1001  
1002 the original neural data in Section A.6 for both the temporally-fixed linear mapping in Figure S3 1002  
1003 (in the “aIT” panel), and with a temporally-varying PLS mapping in Figure S4 (“aIT” in panel 1003  
1004 (a) as well as the data used in panel (b)), all from layer 10 of the 11-layer BaseNet/ConvRNNs. 1004

1005 With these observations, we then proceeded to evaluate the effect of the linear mapping 1005  
1006 on OST correlations in Figure S5. Crucially, in this setting, we train a 100 component PLS 1006  
1007 mapping on the 526 images for which an IT  $d'$  is *not* defined, in order to ensure that the images 1007  
1008 from Section A.8.1 that the OST correlation is evaluated on are not the same images the PLS 1008  
1009 mapping was trained with. 1009

## 1010 Extended Data

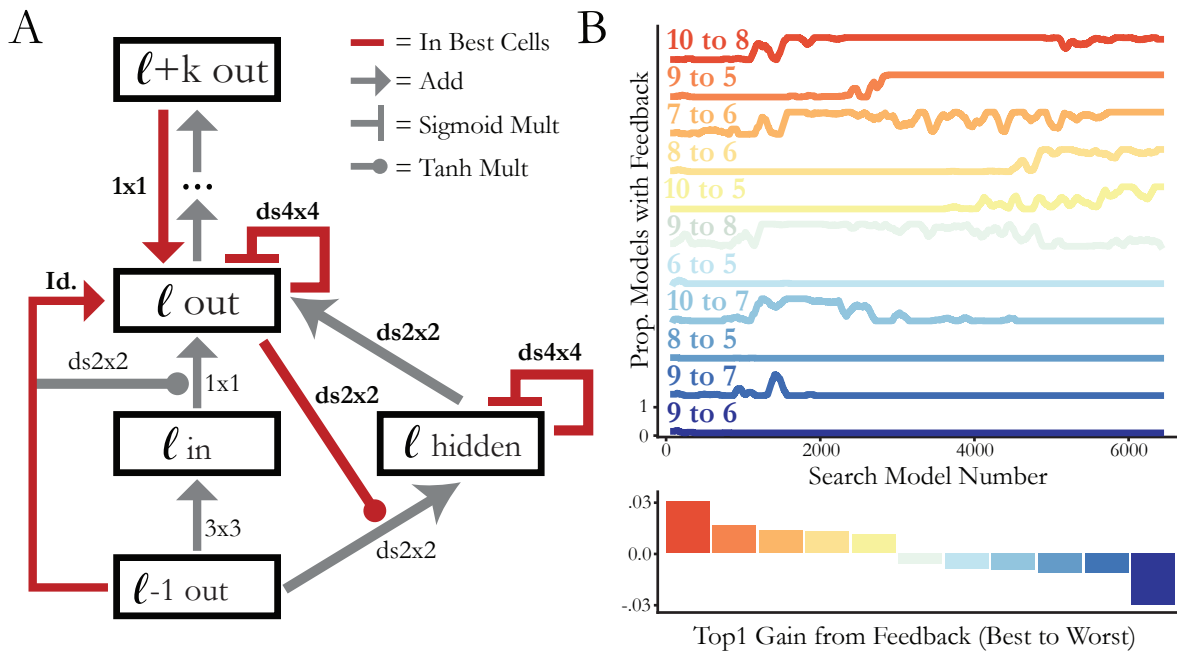
1010

Model	Visual Area	Wilcoxon test $p$ -value
Time Decay	V4	< 0.001
IntersectionRNN	V4	< 0.001
LSTM	V4	< 0.001
UGRNN	V4	< 0.001
GRU	V4	< 0.001
SimpleRNN	V4	< 0.001
RGC Random	V4	< 0.001
RGC Median	V4	< 0.01
Time Decay	pIT	0.022
IntersectionRNN	pIT	< 0.001
LSTM	pIT	< 0.001
UGRNN	pIT	< 0.001
GRU	pIT	< 0.001
SimpleRNN	pIT	< 0.001
RGC Random	pIT	0.31
RGC Median	pIT	< 0.001
Time Decay	aIT	< 0.001
IntersectionRNN	aIT	< 0.001
LSTM	aIT	0.47
UGRNN	aIT	0.09
GRU	aIT	0.16
SimpleRNN	aIT	< 0.001
RGC Random	aIT	< 0.001
RGC Median	aIT	< 0.01

Table 1: Wilcoxon test (with Bonferroni correction)  $p$ -values for comparing each Deeper ConvRNN’s neural predictivity at the “early” timepoints (Figure 3) to the (11-layer) BaseNet.

## 1011 Supplementary Figures

1011



**Figure S1: Optimal local recurrent cell motif and global feedback connectivity. (a) RNN Cell structure from the top-performing search model.** Red lines indicate that this hyperparameter choice (connection and filter size) was chosen in each of the top unique models from the search.  $K \times K$  denotes a convolution and  $dsK \times K$  denotes a depth-separable convolution with filter size  $K \times K$ . **(b) Long-range feedback connections from the search.** (Top) Each trace shows the proportion of models in a 100-sample window that have a particular feedback connection. (Bottom) Each bar indicates the difference between the median performance of models with a given feedback and the median performance of models without that feedback. Colors correspond to the same feedbacks as above.

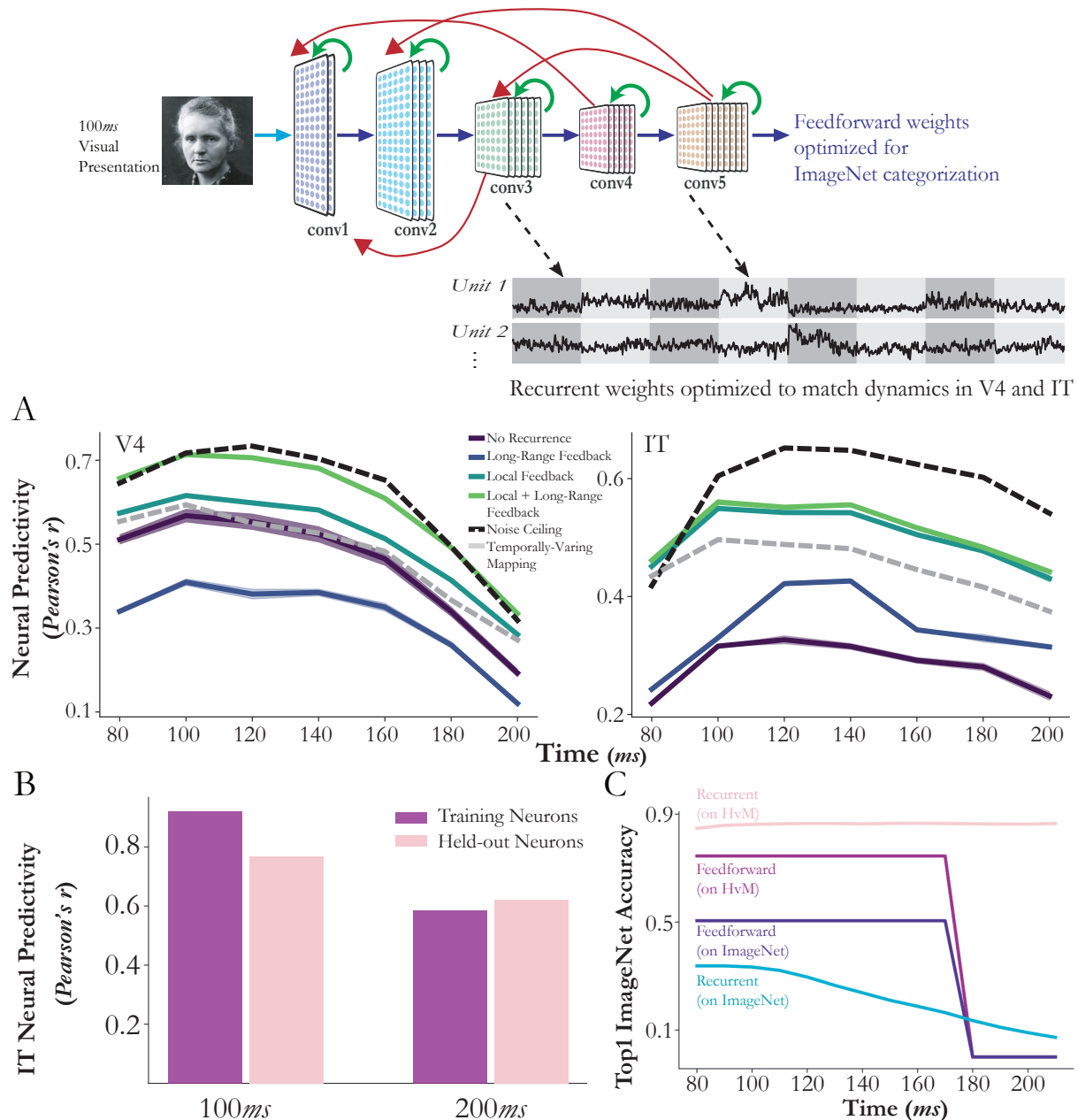


Figure S2: **(a) Both local recurrence and global feedbacks are needed to best fit neural data.** Among a wide range of architectures with different local recurrent motifs and global feedback patterns, the best architecture was one with both gated local recurrence and a global feedback. Local recurrent circuits were particularly useful for improving fits to IT neurons ( $N = 168$ ), whereas both local recurrence and global feedback were critical for improving fits to V4 neurons ( $N = 88$ ). Except for “temporally-varying mapping”, fixed model-unit-to-neuron linear mappings were fixed across all time bins, constraining trajectories to be produced by actual dynamics of the network. In contrast, “temporally-varying mapping” indicates an independent PLS regression for each time bin. The fact that models with local recurrence and global feedbacks are better than “temporally-varying mapping” suggests that some nonlinear dynamics at earlier layers contributed meaningfully to network fits. S.e.m. across four splits of held-out test images. **(b) Held-out neural predictivity.** At both 100ms and 200ms, this direct fitting procedure to the dynamics generalizes to neurons held-out (right bars) in the fitting procedure, a stronger test of generalization than held-out images depicted in the left bars. **(c) Underfitting to the task.** However, a subtle overfitting to the neural image distribution occurs, whereby the task-optimized network whose dynamics are trained on the V4 and IT neural dynamics no longer transfers to ImageNet.

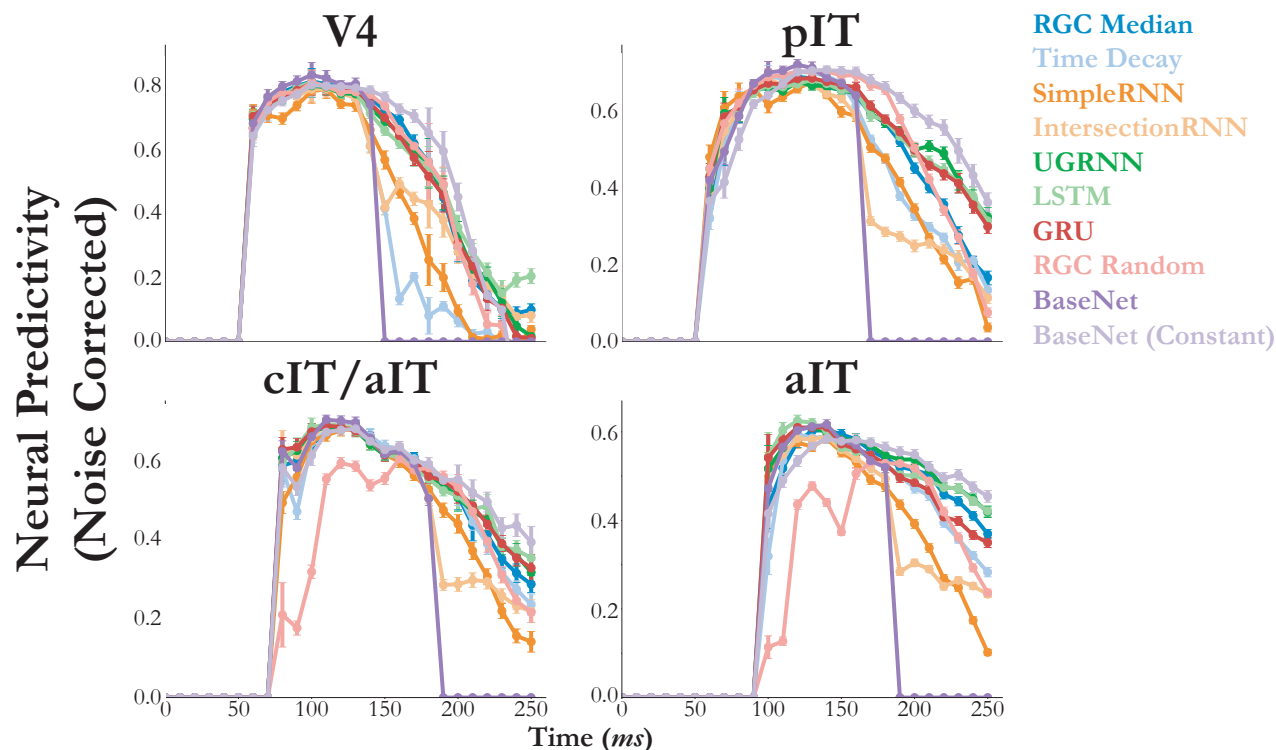


Figure S3: **Modeling primate ventral stream neural dynamics with ConvRNNs.** Fitting model features of ConvRNNs with a temporally-fixed linear mapping to neural dynamics approaches the noise ceiling of these responses in most cases. The  $y$ -axis indicates the median across neurons of the explained variance between predictions and ground-truth responses on held-out images. Error bars indicates the s.e.m across neurons ( $N = 88$  for V4,  $N = 88$  for pIT,  $N = 80$  for cIT/aIT, and  $N = 486$  for aIT). Note that “aIT” refers to a separate neural dataset from primarily anterior IT neurons, detailed in Section A.8.2. As can be seen, the feedforward BaseNet model (purple) is incapable of generating a response beyond the feedforward pass, and certain types of ConvRNN cells added to the feedforward model are less predictive than others.

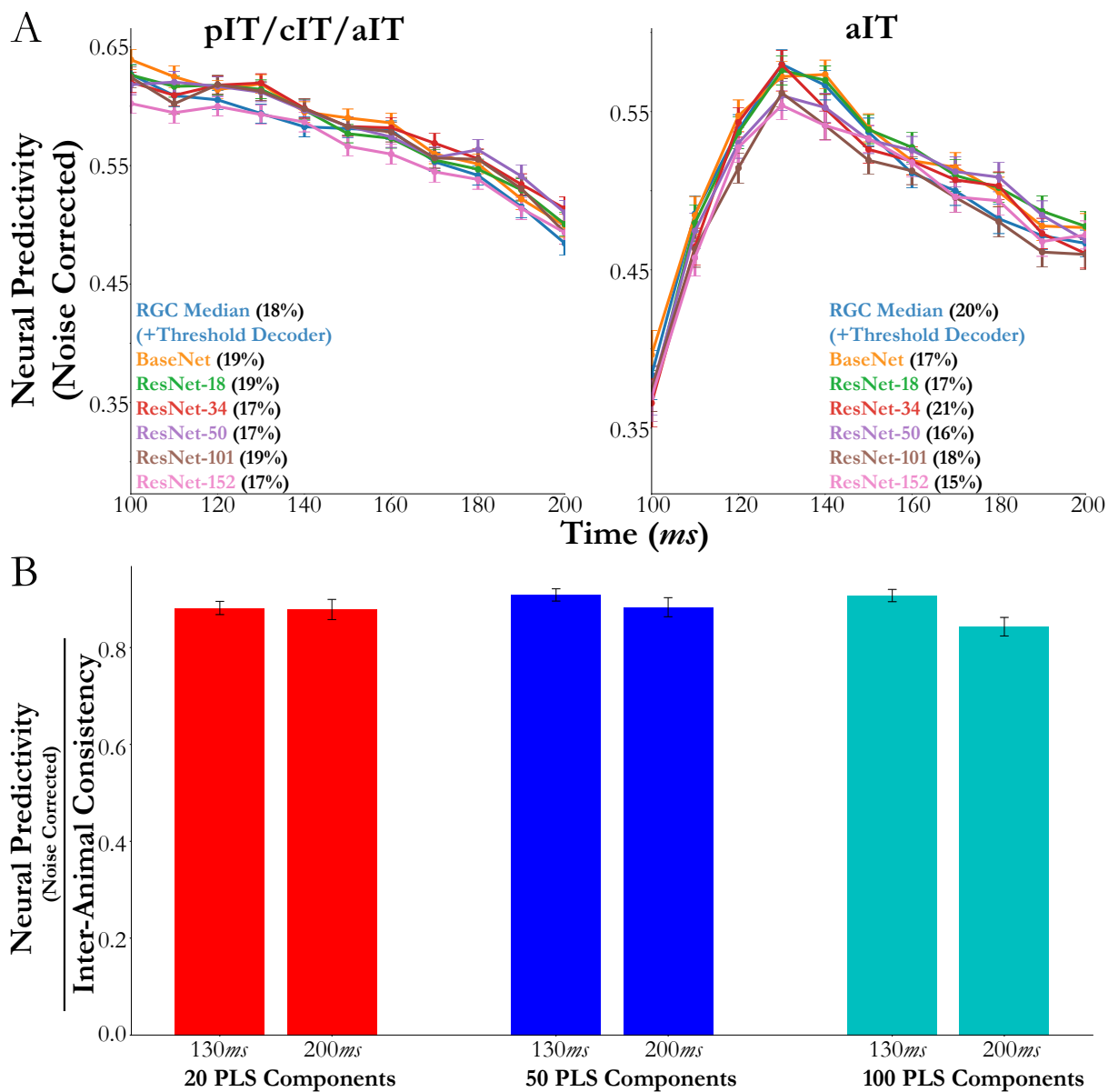


Figure S4: **(a) Increasing feedforward depth does not account for drop in median explained variance from early to late timepoints.** We observe a similar drop in median explained variance from 130-140ms to 200-210ms, between the ConvRNN and deeper feedforward models, where we fix each model's training image size and batch size to be able compare across depths. To compare these two models, we subselect for high reliability neurons (above 0.3 split-half consistency) and use a temporally-varying mapping (PLS 25 components). We plot the median and s.e.m. predictivity in both panels per timebin ( $N = 108, 113, 117, 123, 118, 118, 116, 115, 108, 99, 86$  neurons for each timebin in the "pIT/cIT/aIT" panel, and  $N = 247, 313, 378, 441, 437, 411, 397, 391, 392, 384, 380$  neurons for each timebin in the "aIT" panel). **(b) Drop in explained variance may be exhibited in inter-animal consistency.** Using the neural data described in Section A.8.2, we see a similar inter-animal consistency (metric detailed in Section A.7) at 130-140ms and 200-210ms, as we do with the 11-layer BaseNet. Median and s.e.m. across aIT neurons ( $N = 441$  at 130-140ms and  $N = 380$  at 200-210ms) from the dataset described in Section A.8.2.



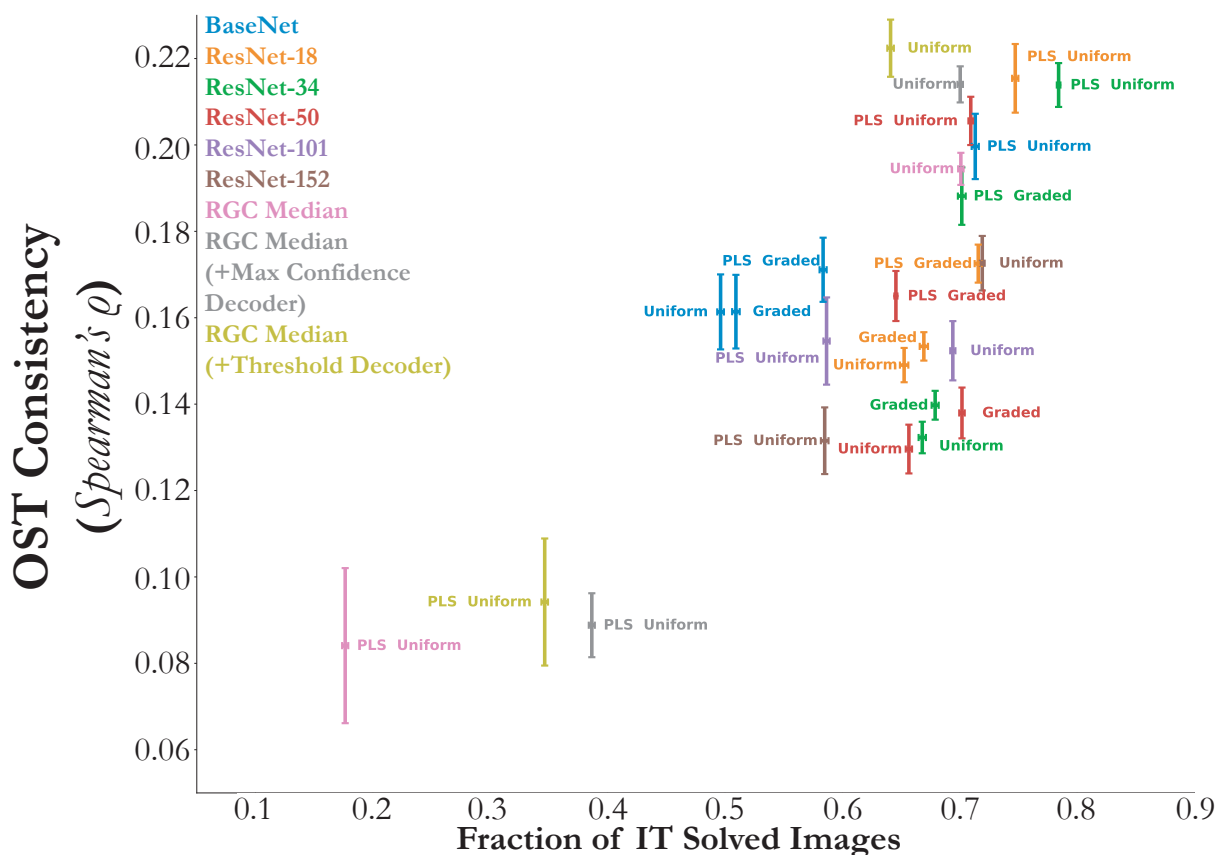


Figure S5: **Behaviorally harmful effect of dimensionality reduction due to linear transform.** Mean and s.e.m. are computed across train/test splits ( $N = 10$ ) when that image (of 1320 images) was a test-set image, with the Spearman correlation computed with the IT solution times across the imageset mutually solved by the given model and IT. As can be seen, a temporally-graded mapping directly from the model features of feedforward models always attains higher OST consistency than a uniform one (“Graded” vs. “Uniform” comparison). We additionally train a 100 component PLS regression to IT responses at each defined model timepoint, where the responses are to a *different* set of images than used to evaluate the OST metric. This procedure, detailed in Section A.8.2, results in an image-computable model on which the OST metric is evaluated on and corresponds to “PLS” prepended to the name of each point on this plot, for any given model and associated temporal mapping. As can be seen, “PLS Uniform” for the BaseNet and ResNet-34 match the OST consistency of the RGC Median ConvRNNs from their original model features. However, “PLS Uniform” for the ConvRNNs and ResNet-101 and ResNet-152 have a significant decrease in OST consistency compared to when evaluated on their original model features, indicating the behaviorally harmful effect of dimensionality reduction due to PLS.