Psychological Review
2014, Vol. 121, No. 2, 248–261

0033-295X/14/$12.00 DOI: 10.1037/a0035943

Neural Networks Learn Highly Selective Representations in Order to Overcome the Superposition Catastrophe

Jeffrey S. Bowers, Ivan I. Vankov, Markus F. Damian, and Colin J. Davis
University of Bristol

A key insight from 50 years of neurophysiology is that some neurons in cortex respond to information in a highly selective manner. Why is this? We argue that selective representations support the coactivation of multiple "things" (e.g., words, objects, faces) in short-term memory, whereas nonselective codes are often unsuitable for this purpose. That is, the coactivation of nonselective codes often results in a blend pattern that is ambiguous; the so-called superposition catastrophe. We show that a recurrent parallel distributed processing network trained to code for multiple words at the same time over the same set of units learns localist letter and word codes, and the number of localist codes scales with the level of the superposition. Given that many cortical systems are required to coactivate multiple things in short-term memory, we suggest that the superposition constraint plays a role in explaining the existence of selective codes in cortex.

Keywords: grandmother cells, localist representations, distributed representations, short-term memory, superposition catastrophe

Supplemental materials: http://dx.doi.org/10.1037/a0035943.supp

There are now many reports of single neurons responding to information in a highly selective manner. This selectivity is best documented in organisms with simple nervous systems (e.g., Elliott & Susswein, 2002), but it is observed in complex organisms as well, including cells in the inferior temporal cortex and hippocampus. For example, a neuron in the hippocampus of a human was found that strongly responded to different photographs of the actress Jennifer Aniston but not to images of other persons, places, or animals (Quian Quiroga, Reddy, Kreiman, Koch, & Fried, 2005). Whether these results are consistent with localist or "grandmother cell" coding is a matter of dispute (Bowers, 2010; Plaut & McClelland, 2010; Quian Quiroga & Kreiman, 2010), but there is no doubt that some neurons respond to information in a highly selective manner (for a detailed review of the neuroscience, see Bowers, 2009).

Given these findings, an important question is why do some neurons respond in this way? A familiar explanation was first advanced by Marr (1971), who argued that memories in the hippocampus are stored in a highly sparse format so that different memories are coded with largely nonoverlapping neurons. This allows a network to learn quickly without new memories interfering with old memories, as needed for episodic memory for example. That is, sparse nonoverlapping memories provide a solution to *catastrophic interference* (McCloskey & Cohen, 1989), or the *stability-plasticity dilemma* (Grossberg, 1980). However, this explanation does not explain the many reports of selective responding of neurons in cortex. For instance, Logothetis, Pauls, and Poggio (1995) trained two rhesus monkeys to identify a large set of novel computer-generated objects. After training, Logothetis et al. recorded from 796 neurons in inferior temporal cortex. A few (3/796; 0.37 %) responded selectively to one object presented from any viewpoint, and, more frequently, neurons (93/796; 11.6 %) responded selectively to a subset of views of one objects but rarely (or not at all) to highly similar objects. This level of selectivity is as great as observed in the hippocampus.

These highly selective responses observed in the cortex require an explanation as well. We propose that the task of activating multiple things at the same time over the same set of units in short-term memory (STM) constitutes a pressure to learn highly selective codes in the cortex, given that various perceptual and cognitive systems within the cortex support STM (Cowan, 2001). On this view, superimposed distributed patterns often result in ambiguous blends that cannot be interpreted, and, in this situation, the only solution is to learn highly selective (or localist) codes, as discussed next.

The Superposition Catastrophe

A number of authors have argued that the superposition catastrophe limits the ability of most networks to coactivate multiple things at the same time over the same set of units (Bowers, 2002; Page, 2000; Rosenblatt, 1961; Von der Malsburg, 1986). According to this hypothesis, a pattern of activation across a set of units in a network can provide an unambiguous representation of a

This article was published Online First February 24, 2014.

Jeffrey S. Bowers, Ivan I. Vankov, Markus F. Damian, and Colin J. Davis, School of Experimental Psychology, University of Bristol, Bristol, United Kingdom.

This research was supported by Leverhulme Grant RJ5538, awarded to Jeffrey S. Bowers.

Correspondence concerning this article should be addressed to Jeffrey S. Bowers, University of Bristol, School of Experimental Psychology, 12a Priory Road, Bristol BS8-1TU, United Kingdom. E-mail: j.bowers@bristol.ac.uk

single item, but superimposing two or more patterns over the same units can result in a blend pattern that is ambiguous in that there is no way to reconstruct the patterns from the blend.

The superposition catastrophe has typically been considered in the domain of vision, and the question has focused on how to bind together features of one object when multiple objects are in the scene at the same time (in order to avoid ambiguous blends of features). For example, Rosenblatt (1961) illustrated the problem in a simple neural network composed of four localist units, with unit 1 responding to a triangle in any position, unit 2 responding to a square in any position, unit 3 responding to arbitrary objects in the upper visual field, and unit 4 responding to arbitrary objects in the lower visual field. As noted by Rosenblatt, this network can unambiguously determine the shape and location of a single object through the coactivation of two units (e.g., a triangle in the upper visual field is coded through the coactivation of units 1 and 3), but it fails to represent the identity and location of two objects when presented at the same time (e.g., a triangle in the upper visual field and a square in the lower visual field will coactivate all four units, and so will a square in the upper visual field and a triangle in the lower visual field).

The superposition catastrophe is also a potential problem in the domain of STM, in which single things (objects, words, etc.) are encoded one at a time but multiple things have to be maintained in memory over time. For instance, the Rosenblatt network would also have difficulty in remembering the following sequence of events: square displayed in the lower visual field followed by a triangle displayed in the upper visual field. The encoding of the first object would be unambiguous, and the item could persist in STM through the continued activation of units 2 and 4. But when the second item is encoded, the same ambiguous blend is produced.

In the above examples the superposition catastrophe occurs in a network with localist representations of shapes and locations. The problem would appear to be more acute when knowledge is coded in a distributed format. For example, consider Figure 1, adapted from Page (2000), which depicts distributed representations for the names John, Paul, George, Ringo, Mick, Keith, Brian, Charlie, Roger, and Pete. Each name is coded as a pattern of activation across 20 units, with four units on and the remaining units off. The identity of the name can be determined by examining the full pattern of activity but not by examining individual units (e.g., activity in the first unit is potentially consistent with Pete, Charlie, or Ringo). The patterns in this example were randomly generated, and this has resulted in some pairs of patterns that are entirely dissimilar (e.g., Pete and Roger are not coded by any common units) and other pairs that are quite similar (e.g., Roger and Brian are coded by three common units). The critical point for our purpose is that blends of these patterns can be highly ambiguous. Consider the patterns for Roger, Brian, and Paul. The two units that disambiguate Roger and Brian (i.e., the unit that is on for Brian and off for Roger, and vice versa) are both on in the pattern that codes Paul. Consequently, the patterns that code the combination of Roger and Paul or Brian and Paul are identical, as can be seen in the bottom two rows of the figure. That is, the resulting blend is ambiguous: It is not possible to determine whether it was produced by combining Roger and Paul or Brian and Paul.

Although Figure 1 shows that a problem with superposition can occur when combining distributed patterns, it should be noted that

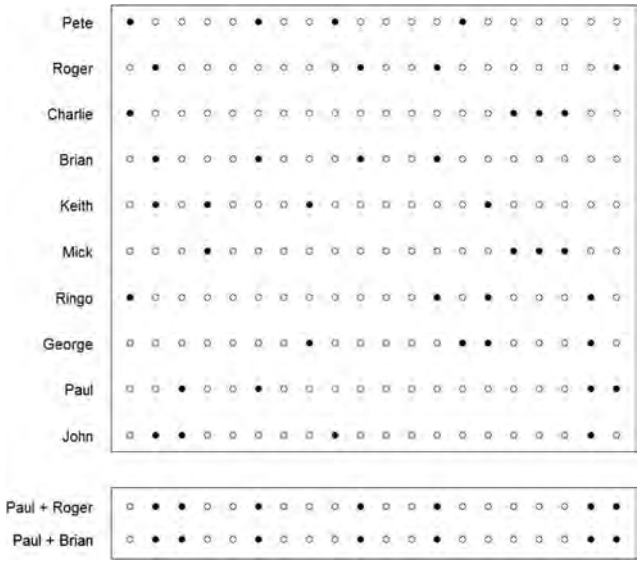

Figure 1. Distributed patterns across 20 units for the names Pete, Roger, Charlie, Brian, Keith, Mick, Ringo, George, Paul, and John, as well as two superpositions (Paul + Roger) and (Paul + Brian) that are ambiguous given that they result in the same blend pattern across the 20 units. The units that are on or off are represented by dark and light circles, respectively.

none of the other possible combinations of two names for this randomly generated set results in this sort of ambiguity. One might therefore wonder to what extent the superposition catastrophe is a genuine problem. To illustrate the superposition catastrophe more formally, we undertook an exhaustive analysis of the particular case in which a set of 20 units is used to code 20 words. As for the above example, we assumed that the patterns are binary (i.e., each unit is either on or off) and that combining two patterns in which a given unit is on in one or both of the patterns results in a blend pattern in which that unit is on. One way to think about this method of combining patterns is to imagine drawing the code for each pattern on a separate overhead transparency; the superposition of a set of words corresponds to the image you see when stacking the corresponding overheads on top of each other. More formally, this method of superposition is equivalent to a logical OR operation. Our approach was to construct vocabularies of 20 words through random sampling and then to determine the superposition corresponding to every possible list of words (up to a maximum list length of six words). Figure 2 shows the proportion of blend patterns that are ambiguous as a function of list length and the sparseness of coding; that is, the number of units that are involved in coding each word. The sparser the input, the less the word patterns overlap; the extreme condition of using only a single unit to uniquely code each word corresponds to a localist representation. Each point is based on an exhaustive sample of lists of a given length. For example, there are 38,760 ways in which six items can be sampled from a vocabulary of 20 words. When four units are used to code each word, 37,042 (96%) of these lists result in ambiguous blends (this figure is the mean of 100 samples, each using a different randomly defined set of patterns for the 20 words in the vocabulary).

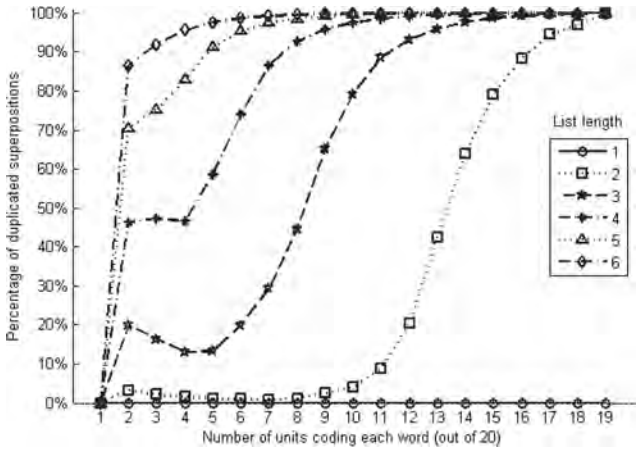

Figure 2. The percentage of ambiguous blend patterns as a function of list length and number of active units when 20 units are used to code 20 words. List length varies from one to six words, and number of active units varies from one unit per word to 19 units per word. Ambiguities increase as a function of both factors, apart from the case of words coded with a single unit (localist coding).

As can be seen in the figure, the degree of ambiguity in superpositions increases as coding becomes less sparse and as the number of patterns being combined (the list length) increases. When each word is coded by four units, combinations of two words are very unlikely to give rise to ambiguous blends; in this respect, the example shown in Figure 1 is somewhat unrepresentative. However, when five words are combined, ambiguous blends are the rule rather than the exception. The problem is worse still when each word is coded by many units (dense distributed coding), but it is apparent even when only two units are used to code each word (sparse distributed coding). The only case in which ambiguous blends are avoided is the localist coding case, in which each word is coded by a single distinct unit.

The results of the analysis depicted in Figure 2 indicate that the superposition catastrophe is a genuine problem for distributed representations, but these results do not imply that specific models that include distributed representations must suffer from this problem. It is conceivable that the problem will not occur in practice in models that have continuous (rather than binary) units and that combine patterns using operations other than the logical OR operator. But it is by no means obvious that this is the case (e.g., averaging continuous units might lead to an even more extreme superposition catastrophe). Our working hypothesis was that the superposition catastrophe would increase in severity as a function of the density of the distributed coding and the number of patterns that must be simultaneously activated. The simulations we present below enabled us to test this hypothesis.[1]

Responses to the Superposition Catastrophe Hypothesis

There have been three general responses to the superposition catastrophe hypothesis. The first (most common) response is to ignore the issue. This is made possible by the fact that most neural networks are designed to code one thing at a time under conditions in which the superposition catastrophe constraint does not arise. Not only does this response leave the problem unresolved, but it may undermine models that succeed with single items. That is, even if a model succeeds with single items, it may solve the problem in a qualitatively different way than humans, who not share this restriction. For instance, a neural network model of word naming that includes distributed representations might account for a range of data on single-word naming, but if the reading system supports the coactivation of multiple orthographic and phonological forms (for the sake of more complex language tasks or STM), the model's solution may mischaracterize the brain's solution even in the restricted domain of single-word naming.

The second response has been to accept that this is an important constraint on perception and cognition and to develop methods to eliminate the problem in neural networks. The solutions typically rely on some sort of localist coding in order to bind together the appropriate features. Consider again the Rosenblatt (1961) example. One solution would be to add an additional layer of localist representations, so-called conjunctive codes, that map together shapes and locations in long-term memory. For example, as noted by Rosenblatt, the coactivation of the localist units *square*, *triangle*, *upper-visual-field*, and *lower-visual-field* is ambiguous (shapes are not bound to location), but the coactivation of conjunctive localist units for *triangle-in-the-upper-visual-field* and *square-in-the-lower-visual-field* is not. A related solution is to introduce dynamic binding, in which localist units are bound together in short-term memory via synchronous firing (e.g., Hummel & Biederman, 1992). To stick with the Rosenblatt (1961) example, one can code the triangle and the square unambiguously if the square unit and a lower field unit are coactive in synchrony, and the triangle and upper-visual-field unit are coactive in synchrony (and out of phase with one another). This solves the superposition catastrophe without adding another layer of conjunctive localist codes (and avoids a combinatorial explosion of localist units). Note, these two approaches should be considered complementary solutions rather than alternatives, with conjunctive codes used for binding together features of familiar things (e.g., binding together four line segments into a localist representation of square), and synchrony binding localist units that together compose less frequent things (binding together the representations of square and lower visual field). For present purposes, the important point to emphasize is that both approaches to binding are motivated by the view that blended distributed patterns are ambiguous, and the solution in both cases is to avoid distributed blend patterns: in the first case by employing local conjunctive codes, and in the second case by activating one thing at a time (through synchrony). Localist codes are a good medium for implementing temporal synchrony (Hummel, 2000).

The third response is to claim that the superposition catastrophe has been overstated. Indeed, advocates of distributed codes can point to existing connectionist models that can coactivate multiple things at the same over the same set of units

[1] It is important to distinguish between the capacity of a network to store multiple items in long-term memory using distributed representations (which is not in doubt) and the capacity of a network to coactivate multiple distributed representations at the same time. The superposition catastrophe hypothesis concerns the later issue.

(e.g., Botvinick & Plaut, 2006; McClelland, St. John, & Taraban, 1989; Touretzky & Hinton, 1988). Botvinick and Plaut (2006) developed a parallel distributed processing (PDP) model of immediate serial recall that coactivated many letters without producing meaningless blend patterns (up to 8 in their simulations, with better performance possible; Bowers, Damian, & Davis, 2009). The model was presented with a list of letters one at a time and trained to reproduce the letters in the same order—a classic test of short-term memory. Strikingly, the Botvinick and Plaut (2006) model not only succeeded but also captured a range of empirical facts about STM. On their view, distributed representations do indeed limit a model's capacity to code for multiple things at the same time, but these limitations help explain human performance.

The successes of existing models clearly show that PDP networks can code for multiple things at the same time over the same set of units. However, it is important to note that these successes do not undermine the potential significance of the superposition catastrophe. The critical question is *how* do PDP models learn to code multiple things at the same time. It is at least possible that PDP models learn highly selective or even localist codes in order to overcome the superposition catastrophe. This might be particularly true when recurrent PDP networks are trained to encode many items at the same time taken from a large vocabulary of items (e.g., Botvinick & Plaut, 2009; Bowers et al., 2009), such that the superposition of distributed patterns is maximally ambiguous.

Below we systematically explore the question of whether a neural network learns highly selective or localist codes in response to the superposition catastrophe. To this end we trained a simple recurrent PDP model to store multiple words at the same time and then carried out "single-unit" recordings on the hidden units—analogous to the single-cell recording studies carried out in neuroscience—to see how the model succeeded. For our purposes, PDP models are useful because they typically learn distributed codes, and, accordingly, they provide a strong test of our hypothesis. Furthermore, it is often claimed that a key advantage of PDP models is that they learn representations that are best suited for a given task (Plaut & McClelland, 2000). That is, the learned representations are emergent rather than "stipulated" by the modeler. So, if a PDP model learns localist codes when coding for multiple things at the same time, this strongly suggests that the superposition constraint provides a pressure to learn selective coding. Such a pressure could not be demonstrated with a more biologically plausible network (e.g., Grossberg, 1980) that was a priori designed to learn local or highly sparse codes.

To explore the impact of the superposition catastrophe per se, we moved away from studying specific cognitive capacities, such as immediate serial recall, that involve more than encoding multiple items at the same time. Rather, we trained a recurrent PDP model to encode a series of words one at a time and then recall them all at the same time. Although this task does not correspond to any behavioral task, it constitutes a relatively pure superposition condition (i.e., the model is simply required to code for multiple items at the same time in its hidden layer). Accordingly, any local codes that develop in this task will likely reflect the impact of the superposition constraint as opposed to any additional computational requirements associated with more complex tasks (such as coding the order of the to-be-remembered items, as required in the serial recall task). Critically, we manipulated the extent to which word patterns were superimposed in order to explore whether PDP models learn more selective codes when the ambiguity associated with the blend patterns is most acute.

Simulation 1

We trained a recurrent PDP model to encode and recall words taken from a vocabulary of 30 words and manipulated the expected severity of the superposition catastrophe in two ways. First, we varied the number of words that the model had to store in STM, with list length varying from 1 to 3, 5, and 7 words. The longer the list, the greater we expected the superposition catastrophe to be. Second, we varied the sparseness of coding in the input layer. That is, we used localist word units (each word was associated with a single input unit), localist letter units (each letter was associated with a single input unit), and distributed letter units (each letter was associated with 3 input units, and each unit was involved in coding 3 different letters). We reasoned that there would be an increase in the superposition across these three input coding schemes, given that there is a corresponding increase in the overlap in the input patterns across these three conditions (as in Figure 2). If the superposition constraint provides a pressure to learn localist codes, then more local codes should be learned in the longer list conditions and more local coding should be learned when the input patterns overlapped more substantially.

The network was composed of 30 input letter units, 200 hidden units, and 30 output word units. Input units were fully interconnected with hidden units, hidden units were fully connected with output units, and the hidden layer was fully recurrent. There was also a bias unit, connected to all the units in the hidden and output layers. When words were coded locally in the input layer, there was one unit devoted to each word. When the words were coded as collections of letters, we organized the input layer into 10 units coding for consonants in the onset position of a word, 10 units coding for vowels, and 10 units coding for consonants in the coda position, with each word coded as one onset, one vowel, and coda unit. The 30 input units coded for the following letters in the onset, vowel, and coda positions, respectively: (*b, c, d, f, g, h, j, k, l, m*) (*a, e, i, o, u, y, aa, ea, ou, oo*) (*n, p, q, r, s, t, v, w, x, z*). Given that each word was defined as the coactivation of one onset, one nucleus, and one code unit, the total number of possible words was 1,000 (10 onsets $\times$ 10 nucleus $\times$ 10 codas). When words were composed of localist letter codes, each word was composed of three active input units (e.g., "ban" was coded by coactivating the input units 1, 11, 21), and when words were composed of distributed letter codes, each word was composed of nine active input units, with each letter coded by three units (e.g., "ban" was coded by coactivating the input units 1, 2, 3, 11, 12, 13, 21, 22, 23). The list of words and their input coding schemes are shown in Table 1. The input layer included a 31st unit that was activated when the list of words was to be recalled. The output units coded for words in a localist manner, with one unit per word.

The training procedure consisted of two phases, encoding and retrieval. In the encoding phase, the network was presented with a series of words, one at a time, that it had to store in the

Table 1
The 30 Words Used in Simulation 1 and Their Corresponding Word, Localist Letter, and Distributed Letter Input Coding Schemes

Word	Localist word input coding	Localist letter input coding	Distributed letter input coding
ban	100000000000000000000000000000	100000000010000000001000000000	111000000011100000001110000000
beq	000000000010000000000000000000	100000000001000000000010000000	111000000001110000000011100000
bis	000000000000000000001000000000	100000000000100000000000100000	111000000000111000000000111000
cep	010000000000000000000000000000	010000000001000000000100000000	011100000001110000000111000000
cir	000000000001000000000000000000	010000000000100000000001000000	011100000000111000000001110000
cot	000000000000000000000100000000	010000000000010000000000010000	011100000000011100000000011100
diq	001000000000000000000000000000	001000000001000000000010000000	001110000000111000000011100000
dos	000000000001000000000000000000	001000000000100000000000100000	001110000000011100000000111000
duv	000000000000000000000010000000	001000000000010000000000001000	001110000000001110000000001110
for	000100000000000000000000000000	000100000000010000000001000000	000111000000011100000001110000
fut	000000000000010000000000000000	000100000000001000000000010000	000111000000001110000000011100
fyw	000000000000000000000001000000	000100000000000100000000000100	000111000000000111000000000111
gaax	000000000000000000000000100000	000010000000000010000000000010	000011100000000011101000000011
gus	000010000000000000000000000000	000010000000001000000000100000	000011100000000111000000111000
gyv	000000000000001000000000000000	000010000000000100000000001000	000011100000000111000000001110
haaw	000000000000001000000000000000	000001000000000010000000000100	000001110000000011100000000111
heaz	000000000000000000000000010000	000001000000000001000000000001	000001110000000001111100000001
hyt	000001000000000000000000000000	000001000000000100000000010000	000001110000000111000000011100
jaav	000000100000000000000000000000	000000100000000010000000001000	000000111000000011100000001110
jeax	000000000000000010000000000000	000000100000000001000000000010	000000111000000001111000000011
joun	000000000000000000000000001000	000000100000000001010000000000	000000111010000000111110000000
keaw	000000010000000000000000000000	000000010000000001000000000100	000000011100000001110000000111
koop	000000000000000000000000000100	000000010000000000010100000000	000000011111000000010111000000
kouz	000000000000000001000000000000	000000010000000000100000000001	000000011110000000111100000001
laq	000000000000000000000000000010	000000001010000000000010000000	100000001111100000000011100000
loon	000000000000000000100000000000	000000001000000000011000000000	100000001111000000011110000000
loux	000000001000000000000000000000	000000001000000000100000000010	100000001110000000111000000011
map	000000000000000000001000000000	000000000110000000001000000000	110000000111000000011100000000
mer	000000000000000000000000000001	000000000101000000000001000000	110000000101110000000001110000
mooz	000000000100000000000000000000	000000000100000000010000000001	110000000111000000011100000001

Note. Each word is coded across 30 input units, with 1 and 0 indicating that the corresponding input unit was on or off, respectively.

recurrent connections in its hidden layer. The second phase consisted of a single step in which the network had to output all of the words at the same time. As a concrete example, consider the task of encoding and recalling the two words "ban" and "bep" in the network with localist letter coding units. The words "ban" (input units 1, 11, 21) and "bep" (input units 1, 12, 22) would be presented in sequence, each stored in the recurrent hidden layer. Then, following the retrieval cue (input unit 31), the model outputs "ban" (unit 1) and "bep" (unit 2) at the same time in the output layer. The same two output units would be simultaneously output when given the sequence "bep-ban" (given that order does not matter).

The network was trained with backpropagation through time, a variant of the backpropagation algorithm that is suitable for training recurrent PDP networks. The learning rate was fixed to 0.01, and a momentum of 0.9 was used. The standard sigmoid activation function was used in both the hidden and the output layer. The gain of the activation function was set to 1. The error at the output layer was computed with the cross entropy function.

Simulation 1 included a total of 12 conditions: 4 × list length (1, 3, 5, and 7 words) and 3 × input-coding scheme (local word, local letter, and distributed letter input coding). When the model was tested in a multiple word condition, all the trained lists contained the corresponding number of words (i.e., when the model was tested on lists of three words, all trained lists contained three words). Given that the difficulty of the task varied across conditions, we report the performance of the model at intervals of 100,000 training trials. In addition, given the slight variability of the results across simulations, we ran each condition 100 times. The average performance of the model is reported in Figure 3.

As is clear from Figure 3, the model needed more training when trained to recall longer lists of words and when trained with the distributed input coding schemes. Still, performance was excellent in all cases by one million trials. The critical question, however, is how did the model succeed across the various conditions? To address this issue, we took a random version of the model (out of 100 runs) after training it for one million trials and recorded the activation of the hidden units in response to words, analogous to the single-cell recording studies carried out in neuroscience. Following training in all of the above conditions we recorded the activation of all 200 hidden units in response to all 30 words (presented one at a time) and displayed the results with a graphical method introduced by Berkeley, Dawson, Medler, Schopflocher, and Hornsby (1995). In this method, a separate scatterplot for each hidden unit is created, and each point in a scatterplot corresponds to a unit's activation in response to a single input (e.g., a word). Level of unit activation is coded along the *x*-axis, and distinct values are assigned to each point along the *y*-axis in order to prevent points from overlapping (words were organized randomly). This method provides a single-unit recording for each hidden unit in response to all words.

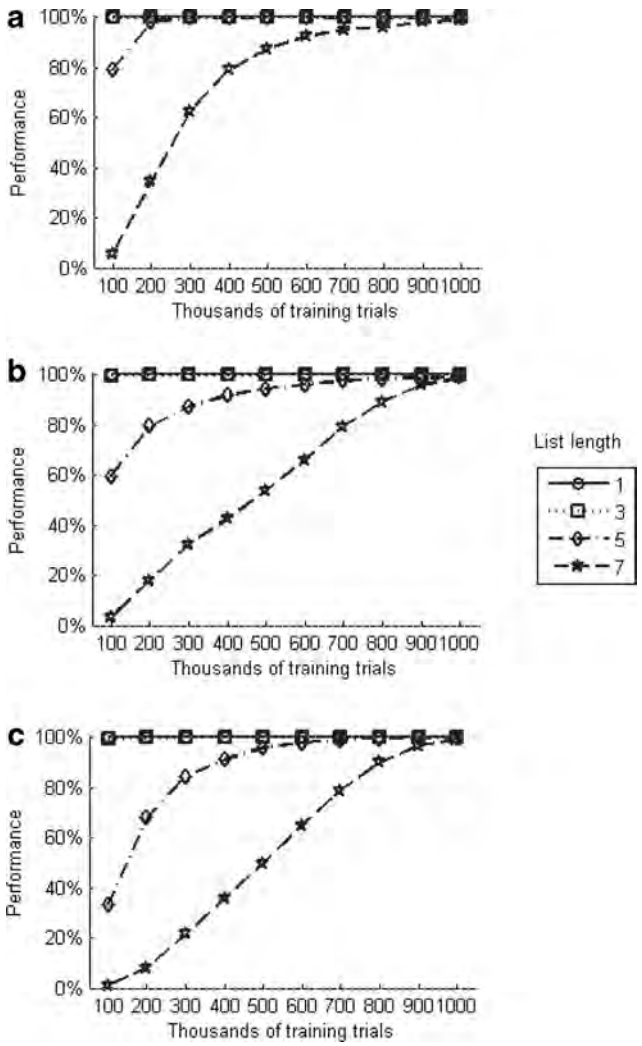

Figure 3. (a) Performance of the network with localist word coding at the input layer as a function of the number of training trials and list length. (b) Performance of the network with localist letter coding at the input layer as a function of the number of training trials and list length. (c) Performance of the network with distributed letter coding at the input layer as a function of the number of training trials and list length.

In Figure 4 we plot the activation of each hidden unit when the model with distributed input letter codes was trained for 1 million trials under two conditions; namely (a) when trained on words one at a time, and (b) when trained on lists of seven words. As is clear from the figure, the pattern of activation across the units is very different across conditions, with selective responding evident only when the model was trained on lists of seven words. This selectivity is highlighted in three specific cases in Figure 4c. For example, Unit 113 was off to all words apart from "gus."

In order to summarize the results more succinctly so that we can display these results across all 12 conditions, we developed a selectivity metric that measured the extent to which a given hidden unit responded to a given word selectively. The selectivity of a unit was computed as the minimal difference in activation between one word and all the rest. These selectivity values can vary from 1 (when a given word drives a given hidden unit to an activation of 1 and all other words lead to no activation; i.e., $1 - 0 = 1$) to -1 (when a given word fails to activate a given hidden unit and all other words drive the unit to an activation of $+1$; i.e., $0 - 1 = -1$). In the latter case, a unit is selectively coding the input pattern by being off. The idiosyncrasies of the modeling architecture, the input set, or the task may have all contributed to the emergence of these OFF selective units, but for present purposes, the important point is that the codes also highlight the computational advantage of selective coding when confronting the superposition catastrophe. In all of the subsequent analyses we treat these OFF units as selective, but the same pattern of results and conclusions follows if only positive selectivity values are considered.

In Figure 5 we depict the selectivity values of the 200 hidden units when the network with a distributed input coding scheme was trained on lists of (a) one, (b) three, (c) five, and (d) seven words. In the last training condition we identified the word to which a given hidden unit responded, with the exact selectivity values shown in parentheses. As can be seen in the figure, the model did not learn any localist codes when trained to recall single words, and it learned localist codes for 21 of the 30 words when trained on lists of seven words at a 0.5 selectivity criterion. Note that the 0.5 criterion is a relatively conservative measure of selectivity. It excludes several units that could reasonably be called selective, such as unit 2, which responds selectively to "diq," but has an absolute selectivity score lower than .5 (see Figure 4c).

In Figure 6 we summarize the number of selective word units across all 12 conditions for which the absolute value of the selectivity measure exceeds 0.5 (averaging across 100 simulations). As is clear from the figure, the number of learned localist word codes increased as a function of the list length and the nature of the input coding scheme, with localist word, localist letter, and distributed letter input coding schemes producing increasingly more localist word codes in the hidden layer. Critically, localist codes very rarely emerged when the model was trained to recall single items; that is, when the model did not confront the superposition catastrophe. These findings are just as predicted on the view that localist coding emerges in response to the superposition constraint, with the number of local codes scaling with the severity of the superposition catastrophe.

Simulation 2

A key feature of the above simulations is that the model included many more hidden units (200) than the size of the trained vocabulary (30 words), and, accordingly, there were more than enough resources for the model to devote a hidden unit to each word. This raises a question. What would happen if the model was trained to recall multiple words taken from a much larger vocabulary, so that the model included fewer hidden units than trained words? This would preclude the use of localist word coding schemes to solve the superposition catastrophe, and, accordingly, if localist word codes were required to perform the task, the model should fail. Alternatively, perhaps the model can adopt a distributed or some other solution under

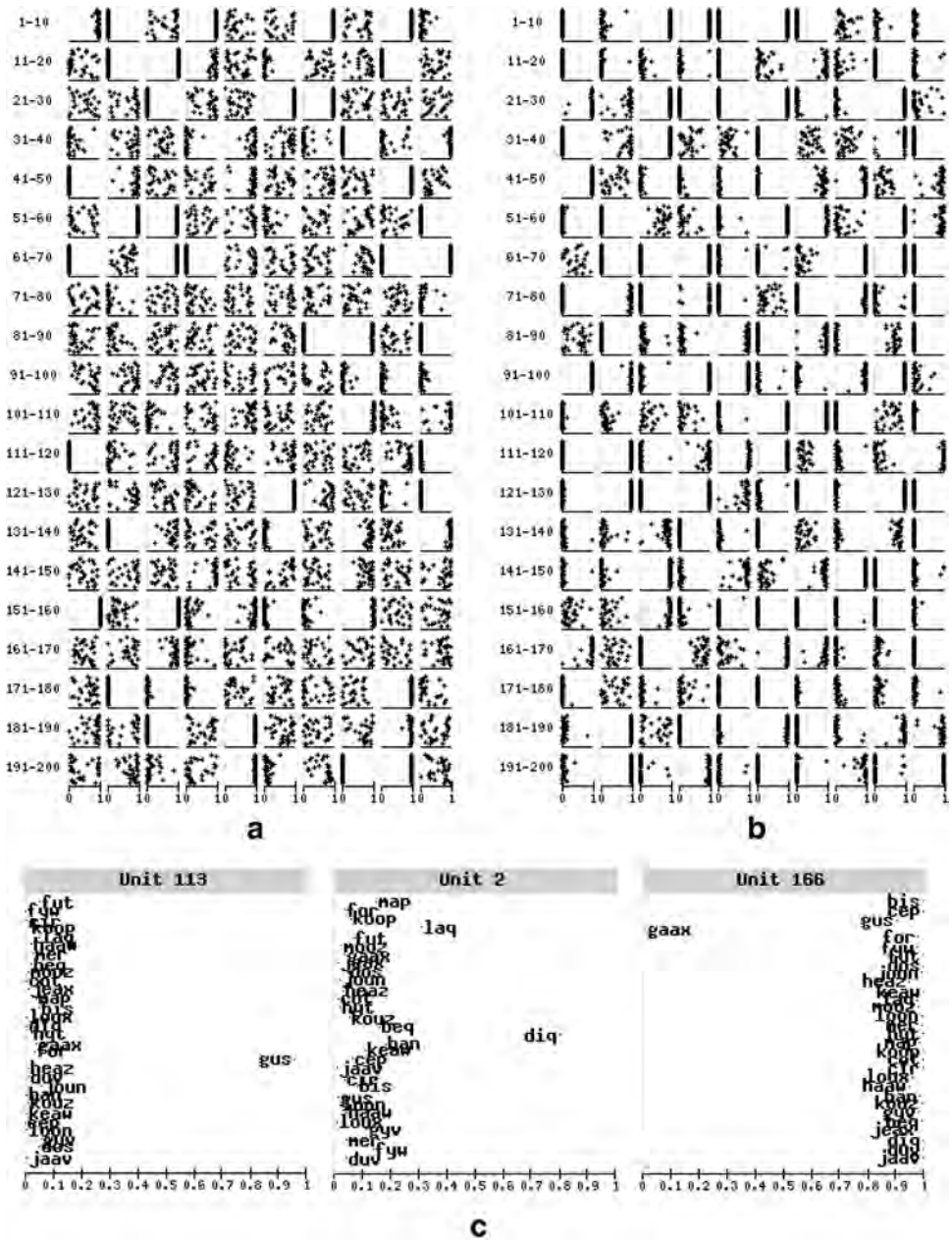

Figure 4. (a) Scatterplots of the 200 hidden units taken from the network with distributed letter coding at the input layer when trained on a vocabulary of 30 words one at a time. Within each scatterplot, each cross represents the unit's response to a particular word. (b) Corresponding plots of the 200 hidden units when network was trained on a vocabulary of 30 words presented in lists of seven words. (c) Labeled scatterplot of Unit 113, Unit 116, and Unit 2 taken from Figure 4b. Unit 113 responds to the word "gus" with a selectivity of 0.9, while Unit 166 responds to the word "gaz" with a selectivity of −0.89. Although Unit 2 responds to the word "diq" more than the other words, its selectivity of 0.49 falls below our threshold of 0.5. As a consequence, it is not considered a selective unit in the analyses that follow.

these conditions and still succeed. In Simulation 2 we explored this issue by training the model to recall words taken from a much larger vocabulary.

We took the model from Simulation 1 with distributed input letter codes and 200 hidden units and trained it on a vocabulary of 300 words. Accordingly, the output layer size was increased from 30 to 300 (one for each localist word unit). Again we trained the model on lists of 1, 3, 5, and 7 words. Given the greater challenge posed by the larger vocabulary, we trained the model for as long as 20 million trials, in order to provide maximum opportunity for the model to succeed. Again, given the slight variability of the results across simulations, we ran

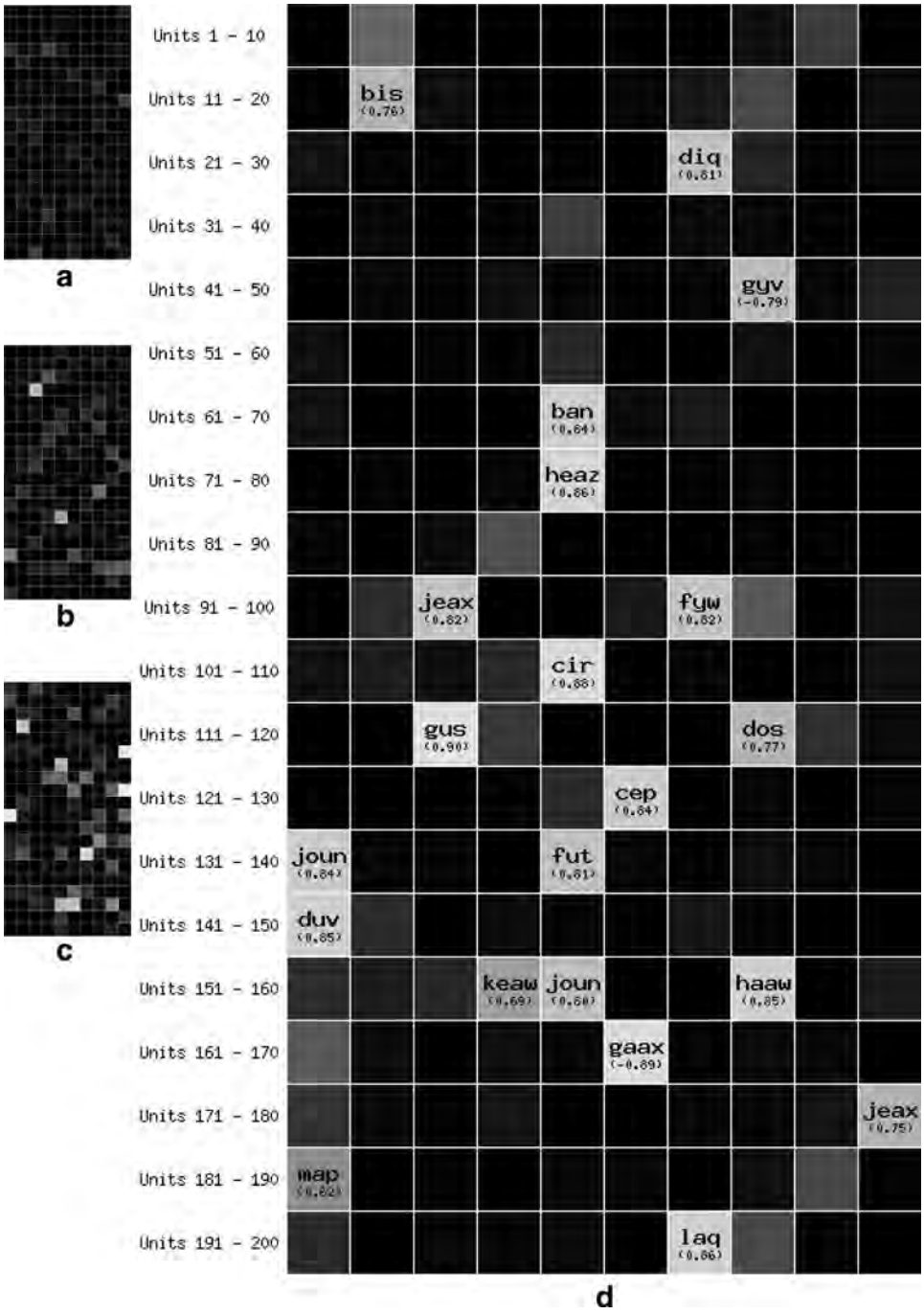

Figure 5. Selectivity plot for the network with distributed letter coding at the input layer trained on (a) single words, (b) lists of three words, (c) lists of five words, and (d) lists of seven words, as a function of vocabulary size. Each hidden unit is coded by a square (10 per row), and degree of selectivity is indicated by the degree of lightness of the square, with light gray referring to a unit with high selectivity and black referring to a unit that is nonselective. In (d), the units that take on selectivity value above .5 are labeled with the letter to which they respond, and the precise selectivity value is presented in parentheses. When trained on words one at a time, all units are nonselective; when trained on lists, some units are selective, with more selective units associated with longer lists.

each condition 5 times, and we report the average performance of the model in Figure 7. As can be seen in Figure 7a, the model did in fact struggle. Indeed, when trained to recall lists of seven words the model reached no better than 20% accuracy. Still, it is interesting to note that by the end of training the model had greater than 95% accuracy on lists of three words and approx-

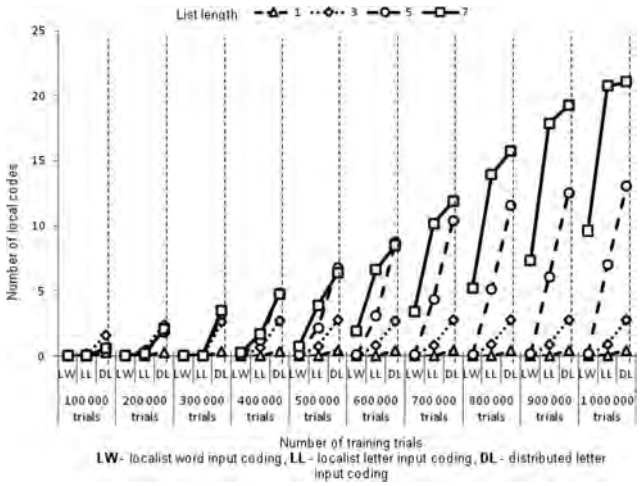

Figure 6. The number of selective word codes in the models with localist word, localist letter, and distributed letter input coding schemes when trained on a vocabulary of 30 words as a function of amount of training and list length.

imately 80% accuracy on lists of 5 words. We report in Figure 7b the cross entropy error that the model was trained to minimize. Here it is evident that performance improved quite early in the training but that the error does not converge to zero when the model was trained on lists of seven words.[2]

In order to gain insight into how the model succeeded (to the extent that it did) across the conditions, we again carried out single-unit recordings from the hidden layer taken from a random version of the model (out of 5 runs) after training it for 20 million trials. In Figure 8 we plot the activation of each hidden unit when the model with distributed letter codes was trained to recall (a) one word and (b) seven words. Once again, no local codes were acquired after training the model on words one at a time. In addition, as can be seen clearly in Figure 8b, there are no units that code for specific words after training on seven words. However, there was a set of units for which there were two discrete bands of activation, such that one subset of words drove the unit to take on one level of activation and another subset of words drove the unit to another level of activation. Furthermore, it is straightforward to interpret many of the bands given that all the words in a given band often contained a specific letter. For instance, consider Unit 60, in Figure 8c, which showed two distinct bands of activation. All the words contained within the highly activated band contained the letter *n*, whereas all the words within the inactive band did not contain the letter *n*. That is, this unit appears to be a localist detector for the letter *n*.

Again, in order to summarize our analyses of the hidden units, we developed a selectivity measure. In this case, the selectivity of a hidden unit was computed as the minimal difference in activation between words that contained a given letter and words that did not contain this letter. In Figure 9 we depict the range of selectivity values across the 200 hidden units when the model was trained on lists of (a) one, (b) three, (c) five, and (d) seven words. In the last case we identify what letter a unit selectively responded to when its absolute selectivity value was above .5, with exact selectivity values shown in parentheses. In Figure 10 we summarize the number of selective letter units across conditions averaging across the 5 runs of the simulation. As can be seen in the figures, the model learned no selective codes when trained to recall single words, learned a few letter codes when trained on lists of three words, and learned many localist letter codes when trained to recall lists of five and seven words. Indeed, when adopting the 0.5 selectivity criterion the model learned almost the full set of possible letter codes when trained in the later conditions. This undoubtedly underestimates the number of selective codes, given that some units, such as Unit 68 in Figure 8c, were selective below this criterion. So, once again, the number of learned localist codes scaled with the level of ambiguity.

Why didn't the localist letter codes support better performance when trained on the longer lists? The answer is straightforward: The coactivated letter codes could not uniquely specify what set of words were presented to the model. To illustrate, consider a case in which the vocabulary of the above model included the words "abc," "def," "ghi," and "adg," and the model is presented the list "abc-def-ghi" to recall. In this situation all the letters for the nonpresented word "adg" are coactivated, and this ambiguity will lead to errors. That is, the model is suffering from the superposition catastrophe despite learning localist letter codes. The longer the list of words to remember, the more ambiguous the blend. This is analogous to the localist version of the superposition catastrophe noted by Rosenblatt and discussed earlier. The solution, as noted by Rosenblatt, is to learn localist codes for complete items, in this case words, but the model did not have the necessary resources. Still, it is clear that learning localist letter codes was better than learning no local codes, and the model did the best that it could with the limited resources at its disposal.

General Discussion

A striking result from neuroscience is that some neurons respond highly selectively to information, both in the hippocampus and in the cortex (Bowers, 2009). There is an ongoing debate as to whether this selectivity is consistent with localist (grandmother cell) coding (cf. Bowers, 2010, 2011; Plaut & McClelland, 2010; Quian Quiroga & Kreiman, 2010), but whatever the case, it is important to determine why some neurons respond in this way.

Our main contribution is to highlight the potential relevance of the superposition catastrophe. The current simulations provide clear evidence that recurrent networks trained to store multiple things (in this case words) at the same time over the same set of units often learn highly selective (indeed localist) representations. Of interest, the constraints posed by the superposition catastrophe in the domain of STM complements the constraints posed by catastrophic interference in the domain of long-term memory

[2] In order to determine whether the limited success of the model on lists of seven words was dependent on the specific learning rate that we employed above, we varied the learning rate parameter from .01 to .02, .03, and .04 and trained the model to up to 20 million trials. The model was limited in its ability to recall seven words at all learning rates (maximum performance in all cases did not exceed 20%), suggesting that the model's performance was restricted by its limited resources rather than the specific learning parameters we employed. In addition, in all cases, the model learned localist codes. This highlights the fact that the emergence of localist codes was not contingent on a specific learning rate. Details of our simulations that varied learning rates can be found at https://sites.google.com/site/superpositioncatastrophe/

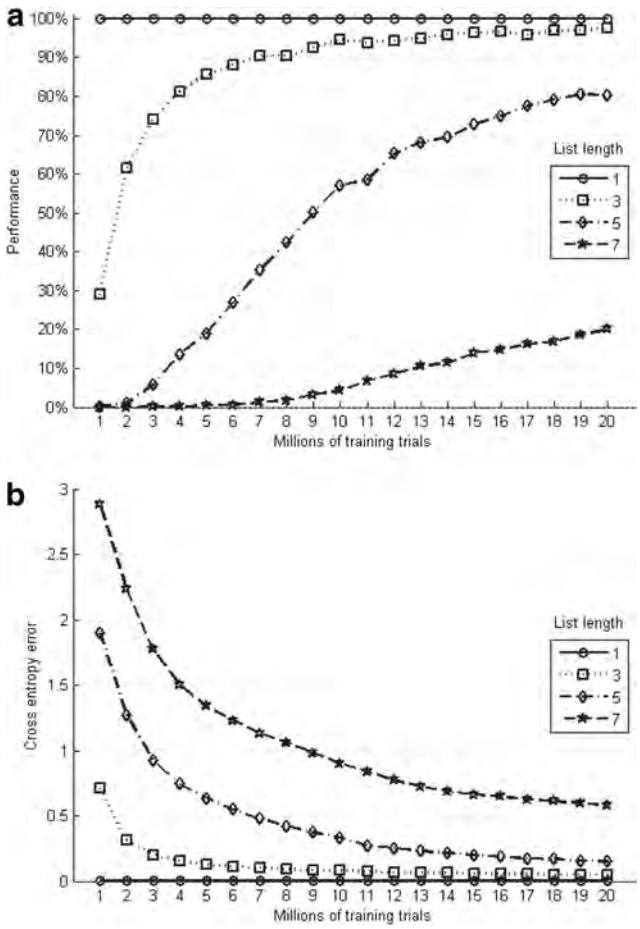

Figure 7. (a) Performance of the model with distributed letter coding scheme and trained on a vocabulary of 300 words as a function of list length and number of training trials. Even after 20 millions of trials, the network performs poorly on long lists. (b) Network error as a function of list length and amount of training. Although the network performed at floor following five million training trials when trained on lists of seven words (see Figure 7a), the network error reduced by approximately half. The error does not converge to zero in this condition.

(LTM). That is, just as distributed representations are a poor medium for STM, distributed representations are a poor medium for rapid learning in LTM (e.g., McCloskey & Cohen, 1989). Together, these constraints on STM and LTM may help explain why neural coding is selective in both the cortex (for the sake of STM) and the hippocampus (for the sake of episodic LTM).

It is important to note that we found that the type of localist coding varied across simulations, with localist word codes learned in Simulation 1 and localist letter codes learned in Simulation 2. This variance reflected the different computational resources available to the models across the two simulations. When the model had more than enough hidden units to encode all the trained words in a localist manner, the model developed localist word codes (see Figure 4c). This was an effective strategy, as it allowed the model to succeed 100% of the time recalling 7 words following a million training trials. However, when there were not enough hidden units to encode words in a localist manner, the model continued to learn localist codes but at the letter level. This solution could not support good performance on the longer lists due to the ambiguities that arise with multiple coactivated letters (the superposition catastrophe). Nevertheless, the results highlight the pressure to learn localist codes in response to the superposition constraint and indicate that localist letters provided a better solution than purely distributed coding.

In one respect, the localist letter coding results in Simulation 2 are the most impressive. That is, the localist letter codes constituted an emergent representation, given that the input layer included distributed letters (each letter was coded as a pattern of activation over three units and each unit was involved in coding three letters) and the output layer included localist word codes (one unit per word). It is sometimes claimed that localist representations are "stipulated" by the modeler (e.g., Plaut & McClelland, 2000), but in this case, the localist letter codes emerged without corresponding input or output representations. This again highlights the computational advantage of localist coding when confronting the task of coding multiple things at the same time.

What should be made of the fact that the models in Simulation 1 and 2 successfully recalled lists of three words relying on relatively few localist codes (with more localist codes emerging only with longer lists)? This finding highlights the fact that distributed codes have some limited capacity to overcome the superposition capacity (as can also be seen in Figure 2). Similar conclusions have been made before. For example, Botvinick and Plaut (2006) argued that their recurrent PDP model of immediate serial recall succeeded on the basis of distributed codes by learning a bias to recall the most likely sequences given its training history. This bias was thought to reduce the ambiguity to such an extent that the distributed representations could support STM at a level commensurate with human performance. However, our findings show that distributed solutions only work under limited conditions. When we increased the severity of the superposition problem, such that the blends of multiple items were highly ambiguous, our models relied much more heavily on localist codes.

In addition, what should be made of the fact that the models never learned localist word codes for all the trained words in Simulation 1? Even in the most difficult training conditions that produced the most localist codes, we only observed approximately 20 out of 30 word codes (at least by our strict standard of .5 selectivity). Does this compromise our claim? Not at all. We are claiming only that the superposition catastrophe provides a pressure to learn selective codes in PDP networks, and this pressure might help explain the many single-cell recording studies that have observed highly selective coding in cortex. Furthermore, we are not claiming that the superposition catastrophe is the only constraint that might contribute to the development of selective coding in cortex. For example, Olshausen and Field (2004); Page (2000); and Thorpe (2011) have all identified key computational advantages of highly selective or localist coding, and it is metabolically expensive to have a high proportion of neurons firing at once (Lennie, 2003). Our main contribution is to provide evidence that the superposition constraint is yet another causal factor that might help explain the repeated observation that some neurons respond to information in a remarkably selective manner.

We want to emphasize again that although our PDP network learned localist codes, we are not committed to the view that the brain relies on localist (grandmother cell) coding. Rather, we take

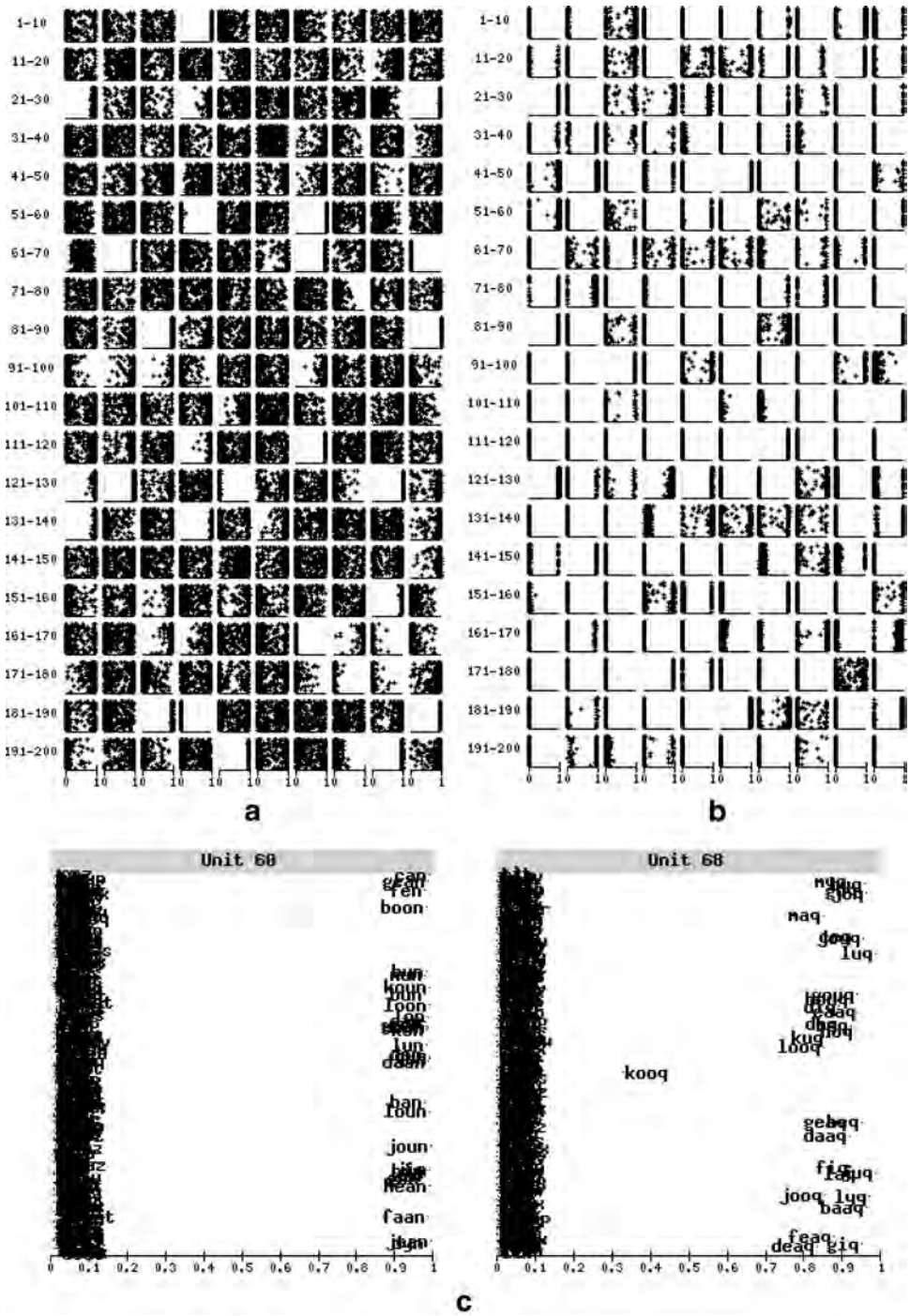

Figure 8. (a) Scatterplots of the 200 hidden units when the network with distributed input letter coding scheme was trained on a vocabulary of 300 words one at a time. Within each scatterplot, each cross represents the unit's response to a particular word. (b) Corresponding scatterplots when the network was trained on lists of seven words. (c) Labeled scatterplot of Unit 60 and Unit 68 taken from Figure 8b. Unit 60 responds to words that contain the letter *n* with a selectivity of 0.80, while Unit 68 responds to words that contain the letter *a* with a selectivity of 0.27. This latter selectivity score falls below our stringent threshold of 0.5, and, as a consequence, the unit is not considered selective in the analyses that follow.

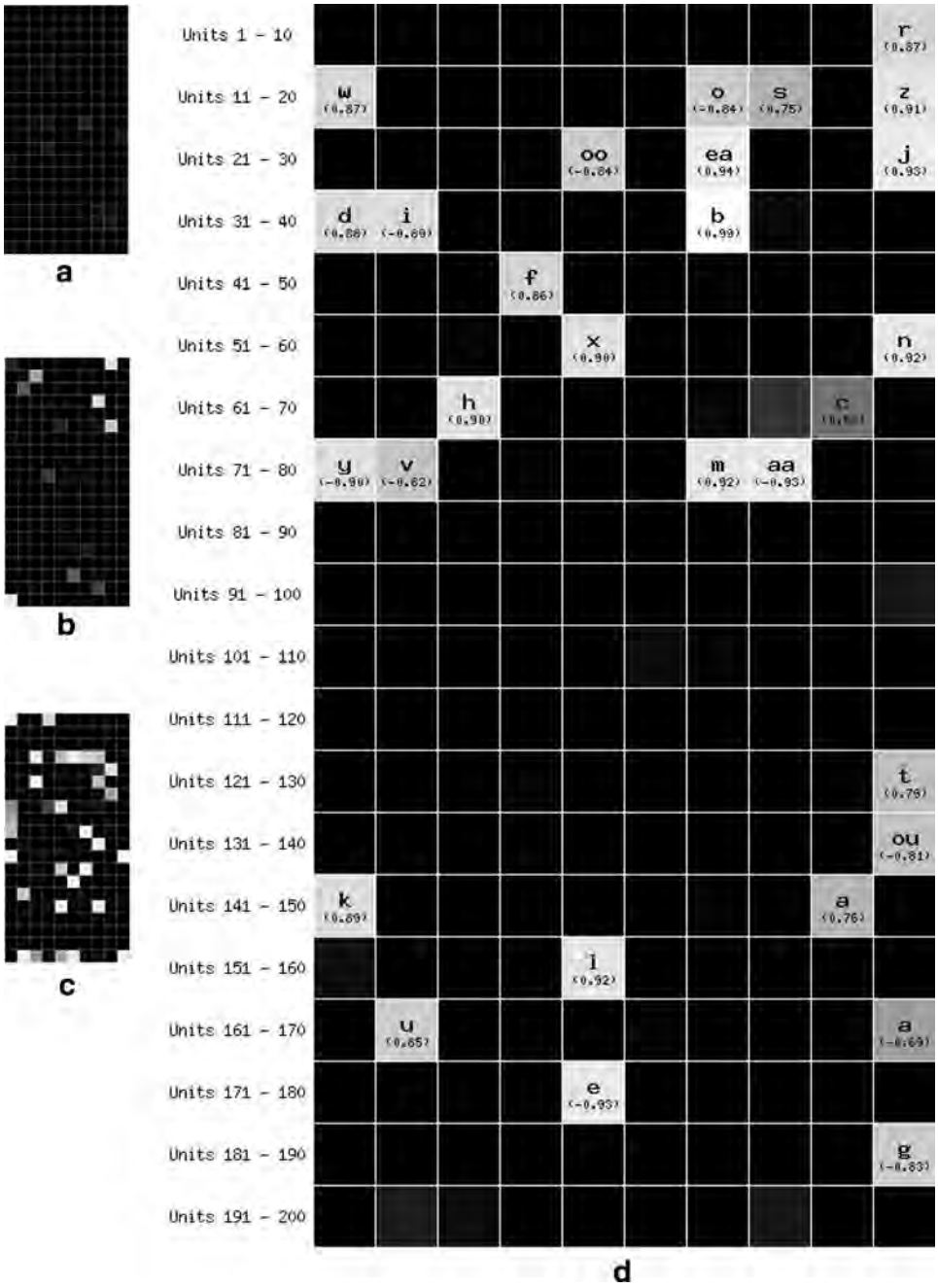

Figure 9. Selectivity plot of the 200 hidden units when the network with distributed input letter coding scheme was trained on a vocabulary of 300 words (a) one at a time, (b) in lists of three words, (c) in lists of five words, and (d) in lists of seven words. Each hidden unit is coded by a square (10 per row), and degree of selectivity is indicated by the degree of lightness of the square, with light gray referring to a unit with high selectivity and black referring to a unit that is nonselective. In (d) the units that take on selectivity value above .5 are labeled with the letter to which they respond, and the precise selectivity value is presented in parentheses. When trained on words one at a time, all units are nonselective; when trained on lists, some units are categorized as selective to letters, with more selective letter units associated with longer lists.

our findings as evidence that the superposition catastrophe provides a pressure to learn highly selective codes. Accordingly, our conclusions are not inconsistent with the claim that the brain relies on populations of highly selective (but not grandmother) neurons to compute (e.g., Pouget, Dayan, & Zemel, 2000). Still, given the combination of computational and biological constraints, this extreme version of selectivity should not be dismissed out of hand.

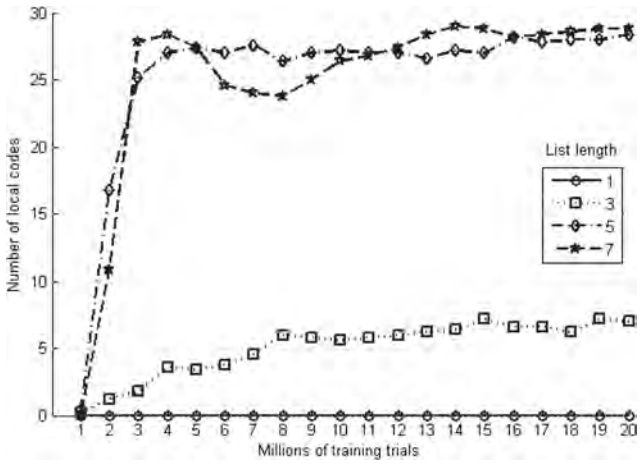

Figure 10. The number of selective local letter codes in the network with distributed input letter coding scheme when trained on a vocabulary of 300 words as a function of amount of training and list length.

As a final note, it is important to emphasize that the current findings do not constitute a challenge for PDP models of perception and cognition. Rather, our findings highlight a problem with how researchers often think about PDP models. On the standard view, PDP models learn distributed codes, and these distributed codes are similar to those learned in the brain. What we have found is that a PDP model learned localist codes when trained to support STM,[3] and, as detailed elsewhere (Bowers, 2009, 2010, 2011), localist codes are at least consistent with what is found in the brain. There are other reasons to think that PDP models are inadequate to the task of explaining cognition and perception and that alternative ("symbolic") network approaches are needed (e.g., Bowers et al., 2009; Hummel & Holyoak, 2003). Nevertheless, the current simulations show how useful PDP models can be in identifying important constraints that all neural networks need to address and one plausible solution to the superposition catastrophe: namely, the development of localist, or highly selective, codes.

[3] We have obtained the same results in PDP models trained on different tasks. For instance, we have found that PDP models of immediate serial recall similar to the models of Botvinick and Plaut (2006) learn localist codes. This further supports our conclusion that localist codes provide a solution to the superposition catastrophe that will arise in any situation in which multiple items are coactive at the same time over the same set of units.

References

Berkeley, I. S. N., Dawson, M. R. W., Medler, D. A., Schopflocher, D. P., & Hornsby, L. (1995). Density plots of hidden unit activations reveal interpretable bands. *Connection Science, 7,* 167–186. doi:10.1080/09540099550039336

Botvinick, M. M., & Plaut, D. C. (2006). Short-term memory for serial order: A recurrent neural network model. *Psychological Review, 113,* 201–233. doi:10.1037/0033-295X.113.2.201

Botvinick, M. M., & Plaut, D. C. (2009). Empirical and computational support for context-dependent representations of serial order: Reply to Bowers, Damian, and Davis (2009). *Psychological Review, 116,* 998–1002. doi:10.1037/a0017113

Bowers, J. S. (2002). Challenging the widespread assumption that connectionism and distributed representations go hand-in-hand. *Cognitive Psychology, 45,* 413–445. doi:10.1016/S0010-0285(02)00506-6

Bowers, J. S. (2009). On the biological plausibility of grandmother cells: Implications for neural network theories in psychology and neuroscience. *Psychological Review, 116,* 220–251. doi:10.1037/a0014462

Bowers, J. S. (2010). More on grandmother cells and the biological implausibility of PDP models of cognition: A reply to Plaut and McClelland (2010) and Quian Quiroga and Kreiman (2010). *Psychological Review, 117,* 300–306. doi:10.1037/a0018047

Bowers, J. S. (2011). What is a grandmother cell? And how would you know if you found one? *Connection Science, 23,* 91–95. doi:10.1080/09540091.2011.568608

Bowers, J. S., Damian, M. F., & Davis, C. J. (2009). A fundamental limitation of the conjunctive codes learned in PDP models of cognition: Comment on Botvinick and Plaut (2006). *Psychological Review, 116,* 986–997. doi:10.1037/a0017097

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24,* 87–114. doi:10.1017/S0140525X01003922

Elliott, C. J. H., & Susswein, A. J. (2002). Comparative neuroethology of feeding control in molluscs. *Journal of Experimental Biology, 205,* 877–896.

Grossberg, S. (1980). How does a brain build a cognitive code? *Psychological Review, 87,* 1–51. doi:10.1037/0033-295X.87.1.1

Hummel, J. E. (2000). Localism as a first step toward symbolic representation. *Behavioral and Brain Sciences, 23,* 480–481. doi:10.1017/S0140525X0036335X

Hummel, J. E., & Biederman, I. (1992). Dynamic binding in a neural network for shape recognition. *Psychological Review, 99,* 480–517. doi:10.1037/0033-295X.99.3.480

Hummel, J. E., & Holyoak, K. J. (2003). A symbolic-connectionist theory of relational inference and generalization. *Psychological Review, 110,* 220–264. doi:10.1037/0033-295X.110.2.220

Lennie, P. (2003). The cost of cortical computation. *Current Biology, 13,* 493–497. doi:10.1016/S0960-9822(03)00135-0

Logothetis, N. K., Pauls, J., & Poggio, T. (1995). Shape representation in the inferior temporal cortex of monkeys. *Current Biology, 5,* 552–563. doi:10.1016/S0960-9822(95)00108-4

Marr, D. (1971). Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences, 262,* 23–81.

McClelland, J. L., St. John, M., & Taraban, R. (1989). Sentence comprehension: A parallel distributed processing approach. *Language and Cognitive Processes, 4,* 287–335. doi:10.1080/01690968908406371

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (Ed.), *The psychology of learning and motivation* (pp. 109–165). New York, NY: Academic Press.

Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology, 14,* 481–487. doi:10.1016/j.conb.2004.07.007

Page, M. P. A. (2000). Connectionist modeling in psychology: A localist manifesto. *Behavioral and Brain Sciences, 23,* 443–467. doi:10.1017/S0140525X00003356

Plaut, D. C., & McClelland, J. L. (2000). Stipulating versus discovering representations. *Behavioral and Brain Sciences, 23,* 489–491. doi:10.1017/S0140525X00473358

Plaut, D. C., & McClelland, J. L. (2010). Locating object knowledge in the brain: Comment on Bowers's (2009) attempt to revive the grandmother cell hypothesis. *Psychological Review, 117,* 284–288. doi:10.1037/a0017101

Pouget, A., Dayan, P., & Zemel, R. (2000). Information processing with population codes. *Nature Reviews Neuroscience, 1,* 125–132. doi:10.1038/35039062

Quian Quiroga, R., & Kreiman, G. (2010). Measuring sparseness in the brain: Comment on Bowers (2009). *Psychological Review, 117,* 291–297. doi:10.1037/a0016917

Quian Quiroga, R., Reddy, L., Kreiman, G., Koch, C., & Fried, I. (2005, June 23). Invariant visual representation by single neurons in the human brain. *Nature, 435,* 1102–1107. doi:10.1038/nature03687

Rosenblatt, F. (1961). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Washington, DC: Spartan Books.

Thorpe, S. J. (2011). Grandmother cells, neocortical dark matter and very long term visual memories. *Journal of Vision, 11*(11), Article 1243. doi:10.1167/11.11.1243

Touretzky, D. S., & Hinton, G. E. (1988). A distributed connectionist production system. *Cognitive Science, 12,* 423–466. doi:10.1207/s15516709cog1203_4

Von der Malsburg, C. (1986). Am I thinking assemblies? In G. Palm & A. Aertsen (Eds.), *Brain theory* (pp. 161–176). Berlin, Germany: Springer.

Received June 28, 2013
Revision received September 17, 2013
Accepted November 4, 2013 ■