Neural networks for word recognition: Is a hidden layer necessary?

Frédéric Dandurand (Frederic.Dandurand@univ-provence.fr)
Laboratoire de Psychologie Cognitive, CNRS, Aix-Marseille University
3, place Victor Hugo, 13331 Marseille, France

Thomas Hannagan (thom.hannagan@gmail.com)
Laboratoire de Sciences Cognitives et Psycholinguistique, EHESS/CNRS/DEC-ENS, École Normale Supérieure
29 rue d'Ulm, 75005 Paris

Jonathan Grainger (jonathan.grainger@univ-provence.fr)
Laboratoire de Psychologie Cognitive, CNRS, Aix-Marseille University
3, place Victor Hugo, 13331 Marseille, France

Abstract

We study neural network models that learn location invariant orthographic representations for printed words. We compare two model architectures: with and without a hidden layer. We find that both architectures succeed in learning the training data and in capturing benchmark phenomena of skilled reading – transposed-letter and relative-position priming. Networks without a hidden layer use a strategy for identifying target words based on the presence of letters in the target word, but where letter contributions are modulated using the interaction between within-word position and within-slot location. This modulation allows networks to factor in some information about letter position, which is sufficient to segregate most anagrams. The hidden layer appears critical for success in a lexical decision task, i.e., sorting words from non-words. Networks with a hidden layer better succeed at correctly rejecting non-words than networks without a hidden layer. The latter tend to over-generalize and confuse non-words for words that share letters.

Keywords: Computational modeling, word recognition, neural networks, reading, priming effects.

Introduction

An important cognitive activity involved in skilled reading is the mapping of retinal images of letters onto abstract word representations. Skilled readers can identify words relatively easily (although not perfectly, see e.g., Rayner, White, Johnson, Liversedge, 2006) even when letter order is jumbled, except for the first and last letters. This suggests that at least one intermediate level of coding exists that abstracts away from absolute letter position and instead codes some information about relative letter order. Such an intermediate level of representation has been studied using a number of techniques including masked priming (see Grainger, 2008 for a review). Robust priming effects found include the transposed-letter effect and the relative-position effect. The transposed-letter effect describes the superior priming observed from primes formed by transposing two of the target's letters (e.g., gadren-garden) compared with primes formed by substituting two of the target's letters (e.g., galsen-garden). The relative-position priming effect describes a processing advantage for targets preceded by primes formed of a subset of the target's letters (e.g., grdn-garden) compared with a prime formed of the same subset of letters in the wrong order (e.g., gdrn-garden).

A number of models have been proposed for an intermediate level of coding that can account for these priming effects (see Grainger, 2008 for a review). Notably, the Grainger and Van Heuven (2003) model of orthographic processing was the inspiration for a computational model that learned to map location-specific letter identities (letters coded as a function of their position in a horizontal array) onto location-invariant lexical representations (Dandurand, Grainger, & Dufau, 2010). Because parsimony dictates to assume a single intermediate level of representation, we considered a neural network architecture with a single hidden layer.

This network architecture with a hidden layer successfully captured transposed-letter and relative-position priming effects (Dandurand et al., 2010). Intermediate representations were explicitly probed and analyzed as patterns of activation at the hidden layer (Hannagan, Dandurand, & Grainger, submitted; see also Plaut, McClelland, Seidenberg, & Patterson 1996 for a discussion of internal representations in neural networks). These patterns were found to have two important characteristics. First, letters seemed to be represented in a semi-location-invariant fashion at the hidden layer. Second, representations at the hidden layer were well-characterized as a holographic overlap coding in which small changes of the inputs resulted in small differences in hidden layer representations. More specifically, differences in patterns of hidden layer activations were monotonically related to differences in identity and position of input letters. For example, patterns of activity at the hidden layer were more different for a two-letter substitution at the input (POLL vs. BULL) than a single letter substitution (PULL vs. BULL) when position in the horizontal array was kept constant. Furthermore, differences in patterns of activity were also larger when the input word was moved by two positions in the alphabetic array (#THAT##### vs. ###THAT###) than moved by a single position (#THAT##### vs. ##THAT####). Holographic overlap coding explains the observed transposed-letter and relative-position priming and

makes a number of predictions which are tested in this article; see (Hannagan et al., submitted) for details.

As they map letters onto words, skilled readers can also perform lexical decision, that is, deciding if a string of letters is a word or a non-word (Meyer & Schvaneveldt, 1971). Lexical decision has been extensively studied, and a number of models exist to account for human performance (e.g., Ratcliff, McKoon, & Gomez, 2004). In the current work, we test our models on a simple lexical decision task, assuming a minimal lexical read-out mechanism, namely that words would activate output units more than non-words. We are not, however, claiming that this ability should be interpreted as a full-blown or realistic model of lexical decision. Note that performing lexical decision is not trivial for networks because non-words are never seen in training as negative evidence, and thus networks may be expected to over-generalize what they consider as words.

In the current study, we revisit the assumption previously made for the need of a hidden layer. We ask if such a hidden layer is required for networks to learn location invariant orthographic representations for printed words. To this effect, we contrast two model architectures: (1) the previous model with a hidden layer and (2) a simpler model without a hidden layer. In this alternative model, letters are mapped to words directly using a layer of connection weights. We compare the two architectures on a number of criteria: (1) their ability to learn the training set, including the anagrams present in the training data, (2) their size and complexity, (3) their capacity to simulate key priming effects, and (4) their capacity to perform a simple lexical decision task. Finally, we investigate how processing and representations differ, how networks without a hidden layer manage to segregate anagrams, and how well these networks conform with the predictions made by holographic overlap coding.

Our goal is to gain insights into the role that the hidden layer plays in performing a word recognition task. Without a hidden layer, networks are computationally limited to taking decisions based on weighted combinations of input letters. It is unclear how, and even if, such model could handle anagrams where the identity of input letters is insufficient to discriminate words, and where position of letters has to be taken into account.

Methods

We compare two architectures of standard multilayer perceptron neural networks. The first one includes a single hidden layer of 91 hidden units with logistic activation functions, identical to (Dandurand et al., 2010). The second one has no hidden layer (inputs are directly connected to outputs). In the two architectures, adjacent layers are fully connected, and are trained using standard backpropagation (learning rate = 0.1, momentum = 0.9) until an SSE of 30. Training material consists of 1179 real words of four letters (same as the one used by McClelland, & Rumelhart, 1988) presented in all 7 possible positions of an alphabetic array (e.g., #ABLE#####, ######ABLE where # are empty, blank slots). Local (sparse) coding is used for input letters (one out of 26 possible letters, for each slot) and output units (one out of 1179 words, also with logistic activation functions). Networks learn to associate letter strings presented at the input with the corresponding output unit coding for some word. For further details, see (Dandurand et al., 2010).

We trained and tested samples of 10 networks for each condition (with and without a hidden layer). Networks varied in the random initial values of their connection weights.

In tests that involve lexical decision, we present some pattern at the input and compute activations of all output units. Output units activated above a threshold value of 0.9 are considered as active, and thus the word associated with the unit as having been detected. For tests that involve priming, a measure dubbed "target supremum measure" (Dandurand et al., 2010) quantifies the ability of some prime to activate the output unit associated with the target word more than any other active output unit[1].

Results

Learning the training set

The training set comprises 1179 words, 24.0% (N = 283) of which are anagrams. Anagrams come in pairs (111 pairs x 2 = 222 words), triplets (15 triplets x 3 = 45 words) and quadruplets (4 quadruplets x 4 = 16 words). These quadruplets (1. live – evil – veil – vile; 2. team – meat – mate – tame; 3. tied – diet – tide – edit; 4. pear – rape – reap – pare) should be especially difficult to discriminate because the same four letters activate four different target word units.

Networks with a hidden layer achieve perfect performance (100%) on the target supremacy measure for the training set. In contrast, networks without a hidden layer reach 98.6%, and more than 95% of anagrams were successfully segregated. In the 1.4% of errors, activations of output units (including the target) fail to reach the threshold of 0.9. These failure-to-recognize errors involved pairs of anagrams (bear – bare, and read – dear) or sets of words from an orthographic neighborhood sharing three letters (bare – mare – pare, seep – seed – deep, and pull – burl – bull).

Model size and complexity

From a size and complexity perspective, the hidden layer adds 91 extra units, and an additional layer of processing. However, in terms of size, networks with a hidden layer actually have fewer connection weights (132 219, i.e., 1179

[1] Models allow for multiple outputs to be activated, but some competitive, winner-takes-all mechanism could be used to select the most active one. Item-level target supremum value was set to 1 when the prime activated the output unit associated with the target lexical item more than any other unit; it was set to 0 otherwise. The target supremum measure of a set of primes was computed as the mean of item-level values for the primes in the set.

outputs x (91 hidden + 1 bias) + 91 hidden x (260 inputs + 1 bias)) than networks without a hidden layer (307 719 connection weights (1179 outputs x (260 inputs + 1 bias)), despite having two layers of weights. We can think of the hidden layer as enforcing data compression from 260 inputs to 91 hidden units, which reduces the number of connections required.

Priming effects

Networks are tested using the relative-position priming and transposed-letter priming manipulations described in (Dandurand et al., 2010). Examples of primes for word ABLE are overlapped on the graphs below, see (Dandurand et al., 2010) for details of the content of testing sets. Primes (e.g., ###ABE####) are expected to activate the target word (e.g., ABLE) more so than any other word, especially when prime letters are in the correct, forward order (ABE) and not the reserved, backward (EBA) order.

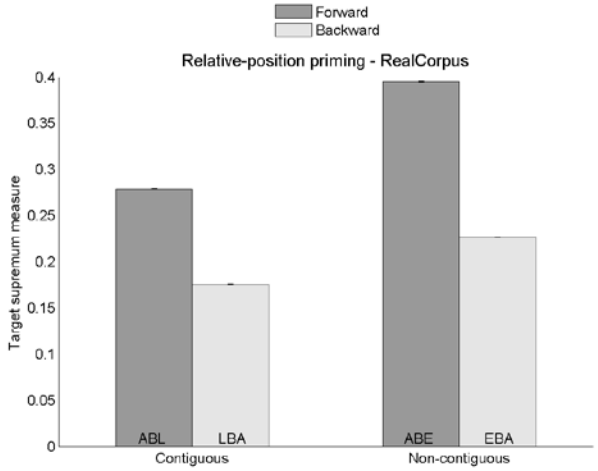

Figure 1 – Target supremum results for the relative-position priming test. Example primes provided for target word ABLE.

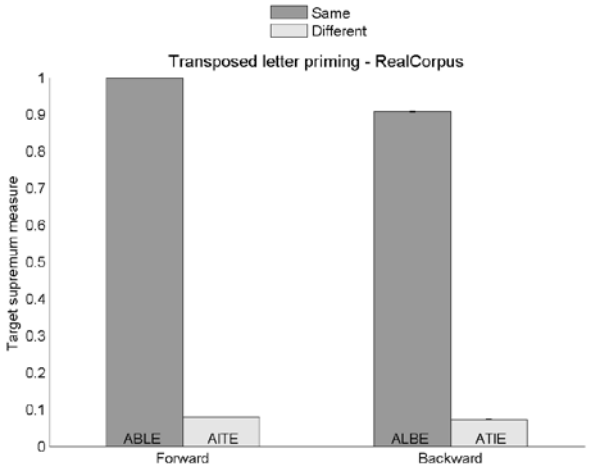

Figure 2 – Target supremum results for the transposed-letter priming test. Example primes provided for target word ABLE.

As we can see, patterns of results are very similar for networks with (see Figures 5 and 6 in Dandurand et al., 2010) and without a hidden layer. More specifically, relative-position primes formed of forward letter subsets yield a higher target supremum measure than backward primes (see Figure 1); and transposed-letter primes containing central letters from the target word yield a larger supremum measure than primes with central letters from a different word (see Figure 2).

Lexical Decision

To test for lexical decision, we assess performance (target supremum measure) on three simple testing conditions: (1) words: all words seen in training in all positions (for a total of 1179x7 patterns); (2) non-words: a sample of 100 patterns made of four random letters presented at a random position in the alphabetic array (e.g. #JKTS#####, ######HIQL, ###BXGA###); (3) letters: a sample of 100 patterns, each made of a randomly selected letter repeated to match word length presented at a random position in the alphabetic array (e.g., ##AAAA####, #####HHHH#). Word patterns are expected to activate, and only activate, their target word unit. We also expect no output word unit to be activated above threshold for patterns in the non-words and in the letters conditions.

Results are shown in Figure 3. As we can see, network with a hidden layer perform much better than networks without one. Networks without a hidden layer are especially poor at correctly rejecting letter patterns, activating several of the words that contain the letter. For example, input pattern ###PPPP### activates 85 word units above threshold including part, open, help, kept, step, post and ship. Similarly, for non-words, errors involve incorrectly activating words that share some letters with the target. For example, input pattern ####KNKR## activates the following word units above threshold: kind, dark, park, mark, link, monk, fork, tank, pork, cork, knot, and trek.

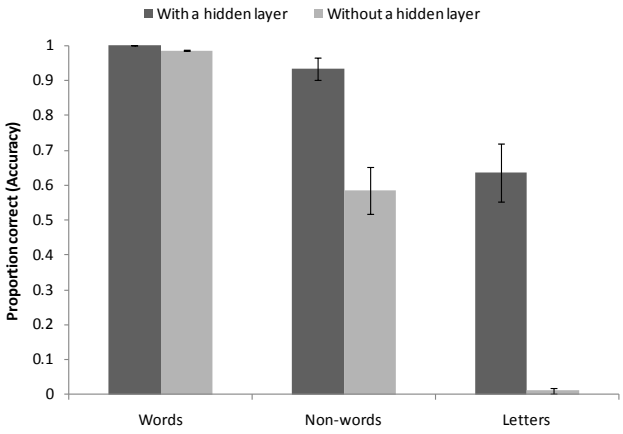

Figure 3 - Accuracy of networks at accepting words, and rejecting non-words and repeated letters.

Discussion

To sum up our results, both networks with and without a hidden layer correctly recognized words at rates reaching 98.6% to 100%. Performance was high even on anagrams (95% to 100%). Both types of networks showed relative-position (see Figure 1) and transposed-letter priming effects (see Figure 2). Networks with a hidden layer are more complex due to the additional hidden units, but contain fewer connection weights. The critical benefit of the hidden layer appears to be in the ability of networks to correctly reject non-words and strings of repeated letters (see Figure 3).

Segregating anagrams

One of the most difficult aspects of the task is arguably that of segregating anagrams. While regular words can be discriminated on the basis of differences of at least one letter, anagram identification must rely solely on the relative position of letters within word. The task appears especially difficult for the network without a hidden layer which is limited to computing linear combinations of independent inputs.

Networks with a hidden layer

In networks with a hidden layer, holographic overlap coding (Hannagan et al., submitted) can explain both transposed-letter priming and the ability of networks to segregate anagrams. During learning, networks form semi-location specific representations for individual letters - assigning similar representations to the same letter input seen at different positions - that is, networks combine letters in a continuous manner to build a string code. Displacing letters (whether in primes or in anagrams) results in small, but measurable differences in patterns of activation at the hidden layer. In the case of transposed-letter priming, most words have no orthographic neighbor, and therefore the target word is still the most activated (e.g., WTIH activates word WITH), and so will be recognized according to the target supremum measure. Networks can capitalize on this small difference in hidden pattern activation to segregate words. It is plausible that this small difference gets enhanced or amplified by the processing of the second layer of weights (hidden to output weights) to generate the correct classification of anagram patterns (e.g., ABLE and BALE as distinct).

Networks without a hidden layer

To gain insights into how networks without a hidden layer can segregate anagrams, we study the connection weights between inputs and outputs after training. The first thing we notice is that connection weights strongly code for the mere presence of letters. Typically, connection weights are small for letters not present in the target word, and large for letters that are present, irrespective of position. For instance, connections weights from input units that code letters A, B, L, and E (in all slots where they have been seen during training) are large to output unit coding for word ABLE. This simple scheme makes each letter vote for the target word, and a word must get 4 votes to be fully activated. This may explain why letters activate very strongly a number of targets, as AAAA also counts as 4 letters of evidence for ABLE. However this does not explain how the network can distinguish between anagrams.

Figure 4 illustrates how networks might manage to segregate anagram patterns. Boxes in the plot show the average magnitude of connection weights between within-word position on the Y axis and within-alphabetic-array (within-slot) location on the X axis for letters relevant to the identification of the target word. For example, for pattern ABLE###### connection weights would be found at boxes (X,Y): A(1,1), B(2,2), L(3,3) and E(4,4); whereas for pattern ###ABLE### relevant boxes would be A(4,1), B(5,2), L(6,3) and E(7,4).

As we can see, there is a negative correlation ($r = -0.73$, $p < 0.01$) in the first within-word position (P) between the average magnitude of connection weights (C) and location (L), while the correlation is positive in the last position ($r = 0.67$, $p < 0.01$). Namely, for the first letter of the word, the connection weight is largest for smaller locations in the slot, and decrease as location in slot increases. This makes intuitive sense, as A######### is better evidence for word ABLE (or any word that begins with letter A) than ######A###, which could be evidence for ######ABLE, but also for ####THAT## or any word having an A in any position. The correlation is reversed for the last slot where say letter E provides more evidence for ABLE if it appears later in the word. The direction reversal suggests an interaction between location (L) and within-word position (P).

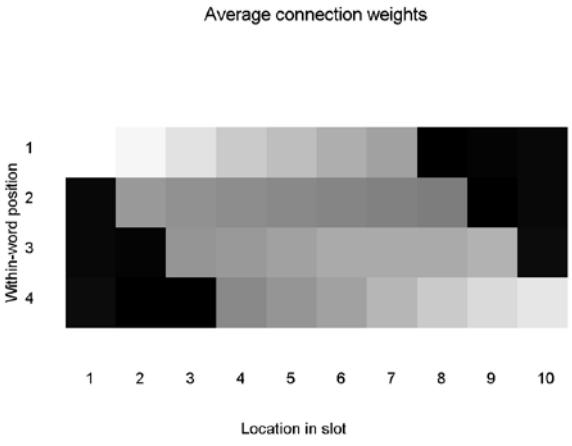

Figure 4 - Average magnitude of weights connecting input units relevant to identifying an output word, by location in the alphabetic array (X axis) and by position with target word (Y axis). Black boxes correspond to positions where letters were never seen in training (e.g., letter A was never seen in slots 8 to 10 for word ABLE, and similarly letter E was never seen in slots 1 to 3).

To test for this interaction, we performed a linear regression with the following model (including LxP to test for interaction effects):

$C = b_0 + b_1L + b_2P + b_3LxP$ (1)

In the fitted model, we get $b_0 = 31.0$ ($p < 0.001$), $b_1 = -4.0$ ($p < 0.001$), $b_2 = -8.9$ ($p < 0.001$) and $b_3 = 62.9$ ($p < 0.001$). This confirms the significant interaction. Redoing the analysis with central locations only (4 to 7), we also get significant coefficients, $b_0 = 22.6$ ($p < 0.001$), $b_1 = -1.8$ ($p < 0.001$), $b_2 = -4.6$ ($p < 0.001$) and $b_3 = 30.4$ ($p < 0.001$).

To sum up, the processing strategy or coding scheme that networks without a hidden layer develop can be described as follows: most important is the number of letters shared between inputs and targets independently of position – we can think of this as input letters providing independent votes for the target words that contain them. The presence of letters is then modulated by the interaction between location and position. This scheme is sufficient to explain how networks can discriminate between anagrams. For instance in strings ABLE and BALE, an equal number of four letter votes go to each word, and connection weights between small slot positions and target word ABLE are slightly larger for letter A than letter B. In contrast, for target word BALE, the connection weight is slightly larger for letter B than letter A. This difference enables the correct target to be activated.

This coding scheme also accounts for the priming effects: larger priming as the number of letters shared between primes and targets increase, and larger priming as the agreement increases between the order of letters in the prime and in the target.

Comparison with holographic overlap coding

How does this processing strategy in networks without a hidden layer compare to holographic overlap coding used by networks with a hidden layer? As mentioned in the introduction, holographic overlap coding makes two important predictions about similarity of activation patterns: a proximity effect and a disruption of activation when replacing letters with other letters of the word (e.g., AAAA for word ABLE). The normalized Euclidian distance between two activation vectors $Act(V_1)$ and $Act(V_2)$ is computed as follows:

$dist = \sqrt{(\Sigma \Sigma (Act(V_{1ij}) - Act(V_{2ij}))^2)} / (N_{pattern} \times N_{activation})$

Activations are taken at the hidden layer, or at the output layer for networks without a hidden layer. The two Σ indicate summing over all patterns and all activation values.

The proximity effect predicts that the Euclidian distance between activation vectors V_1 and V_2 should increase monotonically with the magnitude of displacement of the vectors (i.e., distances). As shown in Table 1, a proximity effect is observed indeed, when vectors V_1 are in the central position (###XXXX###) and vectors V_2 vary in position. Distances presented in the table are normalized using a displacement of 1 as a reference (that is, V_2 ##XXXX#### and ####XXXX##). Vectors V_2 for displacement 2 are #XXXX##### and #####XXXX#; and for displacement 3: XXXX#### and ######XXXX. As we can see, distances increase with displacement, in accordance with the proximity effect.

Table 1: Normalized Euclidian distance for networks with and without a hidden layer, as a function of displacement of letters in the input vector

	Euclidian distance	
Displacement	With hidden	Without hidden
2	1.3	1.5
3	2.2	1.7

Holographic overlap coding also makes a prediction about the effect of letter substitutions: the more letters are replaced, the larger the difference in activation should get. We empirically test this hypothesis by generating samples of 100 test items for which the target word and the location of letters in the input slot is randomly chosen. We compute the Euclidian distance between patterns of activation generated in one of three conditions: (1) transposition – transpose two letters, randomly chosen (e.g., V_1 = ABLE → V_2 = ABEL), (2) one letter substitution with a random letter (e.g., V_1 = ABLE → V_2 = ABWE), (3) one letter substitution with another letter of the target – that is, a letter repetition (e.g., V_1 = ABLE → V_2 = BBLE).

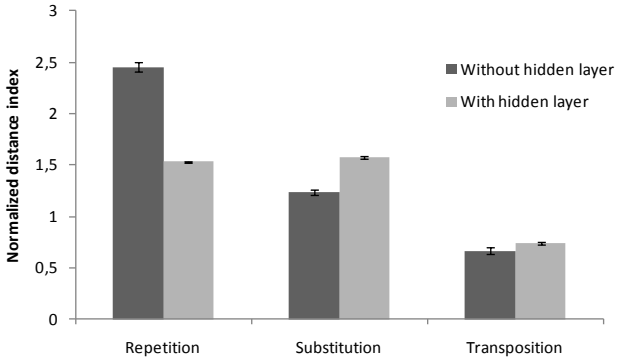

Figure 5: Normalized Euclidian distance index as a function of transformation and architecture type

Holographic overlap coding predicts similar distances for letter repetitions and substitutions, and a lower distance for transpositions. As we see in Figure 5, this is precisely the pattern of distances measured for networks with a hidden layer. However, these predictions are not verified for networks without a hidden layer, namely because distances are too large for the letter repetition set. This somewhat counter-intuitive result can be explained by the fact that repeating a letter means, on average, replacing a letter with a rather frequent letter compared to substituting with a randomly chosen one (as in the substitution case). And thus, many output words activate in the repetition case, which increases the distance due to the higher activation of the non-target words. In sum, we fail to find evidence that networks without a hidden layer implement a holographic overlap coding scheme.

Lexical decision, over-generalization and their theoretical implications

In the lexical decision task, correct rejection of non-words and letters can be interpreted as a test of generalization, which probes the network's ability to correctly set the boundary of word acceptance. Based on a poverty of stimulus argument, we may expect networks to over-generalize, that is being overly liberal in accepting strings as words, because networks see positive evidence for words but never see any negative evidence, i.e., they are never trained to reject non-words. These over-generalization errors are much more common in the network without a hidden layer. This has interesting theoretical implications for the functional role of the hidden layer where independent letters are combined. Given that each letter/position has a uniquely defined code, the network just has to find a way to integrate them so as to ensure that each combination is unique. For instance, using a simple averaging approach, the resulting code for AAAA will be very close to A, in effect providing only evidence for one letter. Without combinations, networks have to base their decisions on some position-weighted voting scheme relating to the presence of letters. This scheme fails to reject non-words cases that consist of 4 repetitions of a letter from the target word.

Beyond simply removing letter duplicates, the hidden layer may well be coding for some letter combination, or sub-lexical units, as postulated in the Grainger and Van Heuven's (2003) model and other models. A simple approach to lexical decision could thus be seen as follows: letters provide evidence for activating sub-lexical units. These sub-lexical units would in turn be combined to activate target words. For non-words, activation of sub-lexical units would be small, and result in activation of output units that fall below threshold.

Conclusion

To summarize, the hidden layer developed a holographic overlap coding scheme which explains priming effects and segregation of anagrams. Because it is sensitive to letter substitutions, this scheme also allows networks with a hidden layer to correctly reject most non-words.

In contrast, networks without a hidden layer have developed a strategy for identifying target words largely based on presence of letters but where letter contributions are modulated using the interaction between within-word position and within-slot location. This modulation allows networks to factor in some information about letter position, which is sufficient to segregate most anagrams, and replicate the previously observed priming effects. On the other hand, these networks are poor at the lexical decision task, as they tend to over-generalize and confuse non-word strings as words. As long as the number of letters is the same and that all input letters exist in the target word, networks do not require that all letters in the target word are present to activate it.

The hidden layer also implements some data compression, by forcing 260 input units to be represented onto 91 hidden units. As a result, networks with a hidden layer have fewer than half the number of connection weights of networks without a hidden layer.

Computational models of word identification are expected to perform well at lexical decision, as humans do. The model with the hidden layer suggests a parsimonious account of lexical decision as an emergent property of the word recognition task (although, again, the setup is highly simplified, and further work would be necessary to fully assess how good of a lexical decision model this is). An alternative explanation consists in using an additional module (performed before, or in parallel with, word identification). For the latter, a network without a hidden layer is sufficient to simply recognize words.

Acknowledgments

This project was supported by the Agence Nationale de la Recherche (grant no. ANR-06-BLAN-0337) and the Europrean Research Council (ERC-230313).

References

Dandurand, F., Grainger, J., & Dufau, S. (2010). Learning location invariant orthographic representations for printed words. Connection Science, 22(1), 25-42. doi:10.1080/09540090903085768

Grainger, J. (2008). Cracking the orthographic code: An introduction. *Language and Cognitive Processes*, *23*(1), 1-35.

Grainger, J., & van Heuven, W. J. B. (2003). Modeling letter position coding in printed word perception. In *The Mental lexicon* (pp. 1-23). New York: Nova Science Publishers.

Hannagan, T., Dandurand, F., & Grainger, J. (submitted). Broken symmetries in a location invariant word recognition network, *Neural Computation*.

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in parallel distributed processing.* Boston, MA: MIT Press.

Meyer, D. E., & Schvaneveldt, R. W. (1971). Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, *90*(2), 227-234.

Plaut, D. C., McClelland, J. L., Seidenberg, M. S., & Patterson, K. (1996). Understanding Normal and Impaired Word Reading: Computational Principles in Quasi-Regular Domains. *Psychological Review, 103*(1), 56-115.

Ratcliff, R., McKoon, G., & Gomez, P. (2004). A Diffusion Model Account of the Lexical Decision Task. *Psychological Review*, *111*(1), 159-182.

Rayner, K., White, S., Johnson, R., Liversedge, S. (2006). Raeding Wrods With Jubmled Lettres; There Is a Cost. *Psychological Science, 17*(3), 192-193