Simulation 1 – Blindsight

General architecture. The architecture of the networks is depicted in Fig. 1. The first-order network was a backpropagation autoassociator, consisting of a 100-unit input layer, itself connected to a layer of 60 hidden units, which were finally connected to a 100-unit output layer. The second-order network was a feedforward backpropagation network, the input of which consisted of a 100-unit comparison matrix, representing the match between the first-order input and output layers. Each of these units calculated the difference between each pair of corresponding input and output units in the first-order network. For instance, if first-order input unit #6(i) had an activation of 0.6 and the first-order output unit #6(o) had an activation of 0.7, the activation of second-order comparison unit #6(c) would be -0.1. It is important to note that these units were thus representing the error of prediction of each first-order output unit. In other words, instead of using this error to drive learning by means of backpropagation (in the first-order network only), the error is represented as an activation pattern, which can be used to drive any secondary task, in this case wagering (in output of the second-order network). The 100 comparison units were connected to two output units representing a high and a low wager. Initial connection weights were between -1.0 and 1.0 for the first-order network and between 0.0 and 0.1 for the second-order output (wager). The comparator weights of the second-order network were set to 1.0 for the connections from the first-order input layer, and to -1.0 for the connections from the first-order output layer.

Wagering specifics. Importantly, the comparator weights between the first-order network and the second-order network (1.0 and -1.0), were not subject to learning. Only the first-order network's weights were modified by learning, as were the second-order network's weights between the comparator units and the wagering units. This allowed for the second-order network to learn to wager at the same time the first-order network learned the discrimination task, while wagering nevertheless remained independent from the particular activation patterns of the first-order network. Instead, the second-order network learned to base its wagers on the degree to which a specific presented pattern (first-order input) corresponded to the internal signal it elicited (first-order output).

Patterns. Network pre-training set consisted of 200 patterns, half of which represented mere noise (unit activations chosen random between 0.0 and 0.02), and half of which represented a possible stimulus (for each pattern, 99 out of 100 units had an activation between 0.0 and 0.02, and one unit had an activation between 0.0 and 1.0. We did not always use an activation of 1.0, as we wanted to create situations in which the network would be unable or have great difficulty distinguishing and localizing the stimulus (subthreshold condition), and hence would learn to wager low in such cases. As the first-order network is contrastive, for "stimulus present" trials the first-order target patterns consisted of 99 units set to 0.0, and one unit set to 1.0, whereas the target patterns for the "stimulus absent" trials consisted of 100 units set to 0.0. A discrimination threshold was applied on the first-order network, so that at least one unit had to be above 0.5 for the pattern to be distinguished from mere noise. The target of the second-order network consisted either of the "high wager" unit set to 1.0 and the "low wager" unit set to 0.0 if (a) the first-order network localized a stimulus (threshold set such that the output unit activation > 0.5) and the stimulus was presented (input contained the stimulus), or (b) the first-order network did not recognize any stimulus and there was none presented; either the second-order network's target was "high wager" = 0.0 and "low wager" = 1.0, if (c) the first-order network recognized a stimulus that was not presented (hallucination), or (d) the first-order network failed to recognize a stimulus that was presented (blindness).

Pre-training. Each network was pre-trained on the 200 patterns for 150 epochs. first-order network's learning rate was 0.9, while second-order network's learning rate (between comparator and wagering units) was set to 0.1. All units' momentum was 0.0, while temperature was 1.0. This pre-training allows the second-order network to learn the degree to which it can trust what the first-order network recognizes. It corresponds to a healthy brain learning to make distinctions between what it does and doesn't see.

Testing. We tested the network in three different conditions. Each of these represented a different way of manipulating the signal-to-noise ratio, and hence a different degree and nature of blindness. First, under "Suprathreshold stimulus" condition, the networks were presented with the same set of 200 patterns as in pre-training. Second, networks were tested under two different blindness conditions. In the "Subthreshold stimulus" condition (representing the Blindsight condition), blindness was simulated by adding noise (+ 0.0012) to every input of the first-order network, except the one representing the stimulus. In the "Low Vision" condition, blindness was simulated by reducing the activation of the stimuli (instead of varying from 0.0 to 1.0, they varied from 0.0 to 0.3).

Additional Results. As shown in Table 3, simulating blindness by reducing signal strength leads to

very different results than by adding noise, and does not result in blindsight. For "Low Vision", we observe a situation unrelated to blindsight but rather reflecting blindness in general, in which the networks completely fail to show any discriminatory ability (50.3% correct) but are still able to wager well above chance (advantageous wagers in 69.6% of the trials). This effect is a consequence of the first-order network's discrimination threshold, which is difficult to attain in this "Low Vision" condition, but which has however no consequence on the spreading of input and output activations to the second-order network. In other words, the first-order network will hardly detect anything, causing the second-order network to lose confidence in the first-order network for the "stimulus present" patterns. Therefore the second-order network wagers low every time it feels a stimulus should have been detected. Thus, although the first-order network does not detect any stimulus, wagers are warranted and wagering performance remains advantageous. Simply said, the metacognitive network "knows" that it is blind.

Robustness of the results. The blindsight simulations are the only ones that depend on the specific choices made for the different parameters (learning rates, epochs, noise), as we wished to reproduce situations of blindsight and of blindness without resulting to more extreme measures such as, for instance, cutting the connections. We included a different possibility of simulating blindness, to illustrate the impact of such choices.

Low Vision	**Correct**	**Incorrect**	**Total**
High Wager	32	12	44
Low Wager	18	38	56
Total	50	50	100

Table 3. Additional results of the Blindsight simulation. Here blindness is simulated by manipulating the signal-to-noise ratio through decreased stimulus activation. Advantageous wagers are underlined.

Simulation 2 – Artificial Grammar Learning Task

General architecture. The architecture was largely similar to that used in the first simulation, and is depicted in Fig. 1. The first-order network was a backpropagation autoassociator, consisting of a 48-unit input layer (representing a string of minimum 3 items and maximum 8, each being one of 5 possible letters, constructed according to the selected artificial grammar), connected to a layer of 40 hidden units, who were connected to the 48-unit output layer. The input of the second-order network consisted of a 48-unit comparison matrix, representing, as in Simulation 1, the difference between the first-order input and output activations. These units were connected to two output units representing a high and a low wager. Initial connection weights were between -1.0 and 1.0 for the first-order network and between 0.0 and 0.1 for the second-order output. The comparator weights of the second-order network were set to 1.0 for the connections from the first-order input layer and to -1.0 for the connections from the first-order output layer.

Wagering specifics. As in Simulation 1, the comparator weights were not subject to learning, for the exact same reasons. The basis of the wagering mechanism was identical.

Patterns. We used 80 random patterns for pre-training. For actual training and testing, we constructed 75 patterns according to a specific Grammar A, and 30 patterns according to a Grammar B, each pattern representing a string from three to eight letters. All input/target patterns consisted of activations of either 1.0 or 0.0. A specific winner-take-all mechanism was added to the output layer in order to select the successive letters of the string that the first-order network recognized. The target of the second-order network was determined in a way identical to Simulation 1.

Pre-training and training. The two network sets were first subjected to pre-training on 80 random patterns for 60 epochs, allowing the second-order network to learn how to wager independently of any first-order task. In order to achieve this, half of the patterns were accompanied by learning in the first-order network, while the other half were not. In both cases, the second-order network had to wager high when first-order input and output matched, and low when they did not. Following pre-training, all first-order network's connections were reset to initial conditions, whereas second-order network's weights were kept as was until the end of the simulation. During the actual training phase, only the first-order networks were trained again on 45 patterns of Grammar A. Of 30 networks, 15 were assigned to a "High Consciousness" condition and were trained for 12 epochs, while 15 networks in the "Low Consciousness" condition were

trained for only 3 epochs. During their respective periods of training, first-order network's and second-order network's learning rates were set to 0.4, units' momentum was 0.5, and temperature was 1.0. The use of different learning phases for the first-order network and the second-order network in this simulation provides an illustration of first-order and second-order independency in the brain.

Testing. We tested all 30 networks on the same set of 60 patterns, consisting of 30 novel Grammar A patterns, and 30 Grammar B patterns.

Robustness of the results. The simulations do not depend on the specific choices made for the different parameters (learning rates, momentums, epochs), but should be manipulated one by one in order to maintain the generalization effect on the second-order knowledge. For illustrative purposes only, Table 4 lists a detailed breakdown of the simulation results. We see that in the case of the simulation, networks were never incorrect for non-grammatical strings. Additional research demonstrated that the use of different parameters for the simulation would allow the occurrence of such classification mistakes, although, without the corresponding data from Persaud et al., we could not look for matching results.

Discrimination		**Experiment**				**Simulation**			
Implicit	**Correct**	**Incorrect**	**Total**	**Correct**		**Incorrect**		**Total**	
Classification	**NA**	**NA**	**NA**	**G**	**NG**	**G**	**NG**	**G**	**NG**
High Wager	36	6.5	42.5	35.1	37.6	17.6	0	52.7	37.6
Low Wager	44.5	13	57.5	8.4	62.4	38.9	0	47.3	62.4
Total	80.5	19.5	100	43.5	100	56.5	0	100	100
Explicit	**Correct**	**Incorrect**	**Total**	**Correct**		**Incorrect**		**Total**	
High Wager	53.2	7.3	60.5	74	52.9	0.7	0	74.7	52.9
Low Wager	20.1	19.4	39.5	22.4	47.1	2.9	0	25.3	47.1
Total	73.3	26.7	100	96.4	100	3.6	0	100	100

Table 4. Breakdown of the AGL simulation results. Each cell for the simulation data represents a breakdown of the results for grammatical (G) and non-grammatical strings (NG). The results in Table 2 represent the average of these two numbers. For Persaud et al.'s experimental data, no such breakdown was available for comparison (NA). Advantageous wagers are underlined.

Simulation 3 – Iowa Gambling Task

General architecture. The architecture differed substantially from that of the first two simulations. We used two connected feedforward backpropagation networks (Fig. 2). The first-order network consisted of 5 input units, representing the last selected deck (1.0 for the corresponding unit and 0.0 to the 3 others) as well as the last obtained outcome (win = 1.0 ; loss = 0.0). These input units were connected to a set of 40 hidden units, which were connected to 4 output units representing the four decks. The 40 first-order network's hidden units also served as input units to another set of 40 second-order network's hidden units. These latter units were then connected to 2 output units representing a high or a low wager. Initial connection weights were between -1.0 and 1.0 for the first-order network and the second-order network. We added ± 0.02 noise to all activations to allow for card deck exploration from the beginning of the training phase.

Wagering specifics. All network weights were subject to learning, meaning that the second-order network would learn, based on the information contained in the first-order network's hidden units, about how successful past performance had been.

Patterns. In our implemented version of the Iowa Gambling Task, two of the decks were coupled with a win in 70% of the trials, while the two others only in 30%. For any given trial at time t, the first-order network's input would consist of a contrasted version of the first-order network's output pattern at t-1 (strongest activated unit was selected by winner-take-all), together with the corresponding win or loss. Although the input patterns reflect the reality of the experiment, it is in fact of no importance since the solving of the task is only based on exploration. The target of the first-order network was set such that, in case of reward, the output corresponding to the selected deck was 1.0 and the others 0.0, and in case of loss,

the selected output was 0.0 and the others 1.0. This method encouraged exploration. In parallel, the target of the second-order network consisted either of the "high wager" unit set to 1.0 and the "low wager" unit set to 0.0 if the first-order network selected a winning card; or the second-order network's target was "high wager" = 0.0 and "low wager" = 1.0 if the first-order network selected a losing card. Thus we ensured that the properties of the four decks were not explicitly revealed whereas exploration would elicit the expected wagering strategy.

Training. 30 networks were trained for 2000 trials (200 epochs) under two conditions, each involving a different degree of awareness. The first-order network learning rate was set at 0.002 for both groups of 15 networks, while the second-order network's learning rate was 0.0003 in the "Low Consciousness" condition, and 0.015 in the "High Consciousness" condition.

Robustness of the results. The simulations do not depend on the specific choices made for the different parameters (learning rates, decks' probabilities, epochs, noise), and may all be manipulated at the same time. However, substantial noise is necessary for inducing the networks' initial exploration of the card decks.