The Neural Database

Gary J. Lassiter

Artificial neural networks do not explain one of the most obvious capabilities of the human brain: the speed with which we learn and utilize new information. We do not, for example, learn to recognize the various breeds of dogs by training ourselves with thousands of images in a dedicated training session. We learn about dogs one dog at a time, accumulating knowledge over our lifetimes, the same way a dynamic database accumulates information.

This document presents an alternate method of implementing artificial intelligence: the Neural Database (Ndb). Individual neurons in an Ndb represent individual items of data and only connect to other neurons which share that data. A Neural Database is not trained; it is taught by simply adding data. The intelligence in the system is "built in" before any data is added. When an inquiry is made into an Ndb, some or all of the neurons are activated and the database's intelligence engine, the Simple Competitive Unit (SCU), runs a series of neuron competitions to determine the system's response to the inquiry. The SCU conducts competitions by launching a variety of excitatory and/or inhibitory spike trains through the connections between contending neurons.

Data Storage in a Neural Database

When data is added to an Ndb, an Output Neuron (ON) is created to represent the data. One or more Recognition Neurons (RNs) will also be created, if they do not already exist, to represent each element of the ON's identifying information - its Recognition List of RNs (see Fig. 1). For example, in an Ndb designed to recognize words in text, each word in the database is an ON, each letter is an RN, and the spelling of the word is the ON's Recognition List.

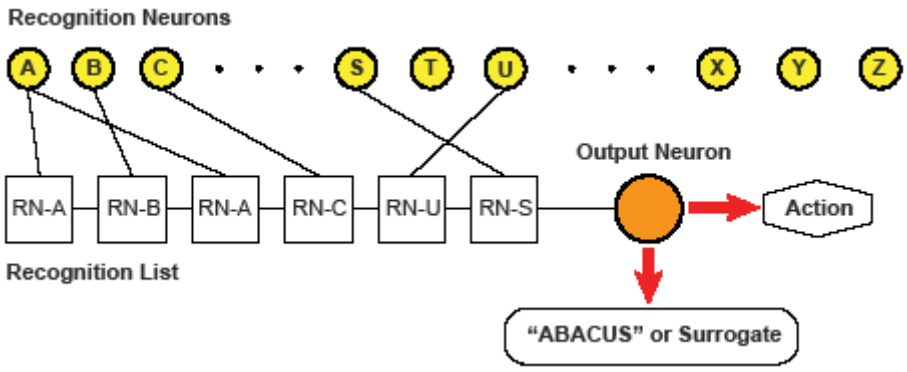

Fig. 1. The word ABACUS stored in a Neural Database.

Every RN maintains a list of every ON in which it appears along with its position(s) in the ON's Recognition List. Also, as shown in Fig. 1, an ON may have an Action associated with it as well as a Surrogate. Whenever an ON appears in the result to an inquiry, the Action, if it exists, will be executed and the Surrogate, if it exists, will be reported instead of the ON's stored data.

Processing an Inquiry

Listed below, and diagramed in Fig. 2, are the six main processing steps used by the Ndb to resolve an inquiry.

1. Preprocessor	Convert the inquiry into a list of RNs, referred to as the Input Stream. Anything in the inquiry which cannot be matched with an RN will be ignored.
2. Scanner	Scan the Input Stream RNs to identify, or "recognize," all the ONs to which they are connected.
3. Remove Weak Ons	Cull from the list of recognized ONs those which do not have enough RNs from the Input Stream. Also remove any ONs which have too many out-of-order RNs.
4. Adjust ON Boundaries	Each recognized ON has boundaries which define its territory in the Input Stream. Conflicts among these territories can result in shrinking some boundaries, expanding others, and in some cases eliminating weakened or enveloped ONs.
5. Assemble Branches	Generate a list of groups of ONs in which the end boundary of one ON is followed by the begin boundary of the next ON. Each group, referred to as a "branch," should span as much of the extent of the Input Stream as possible.
6. SCU Single Elimination Tournament	Pair up the branches to run one-on-one competitions in the SCU. Winners advance to the next round until the final winner is determined.

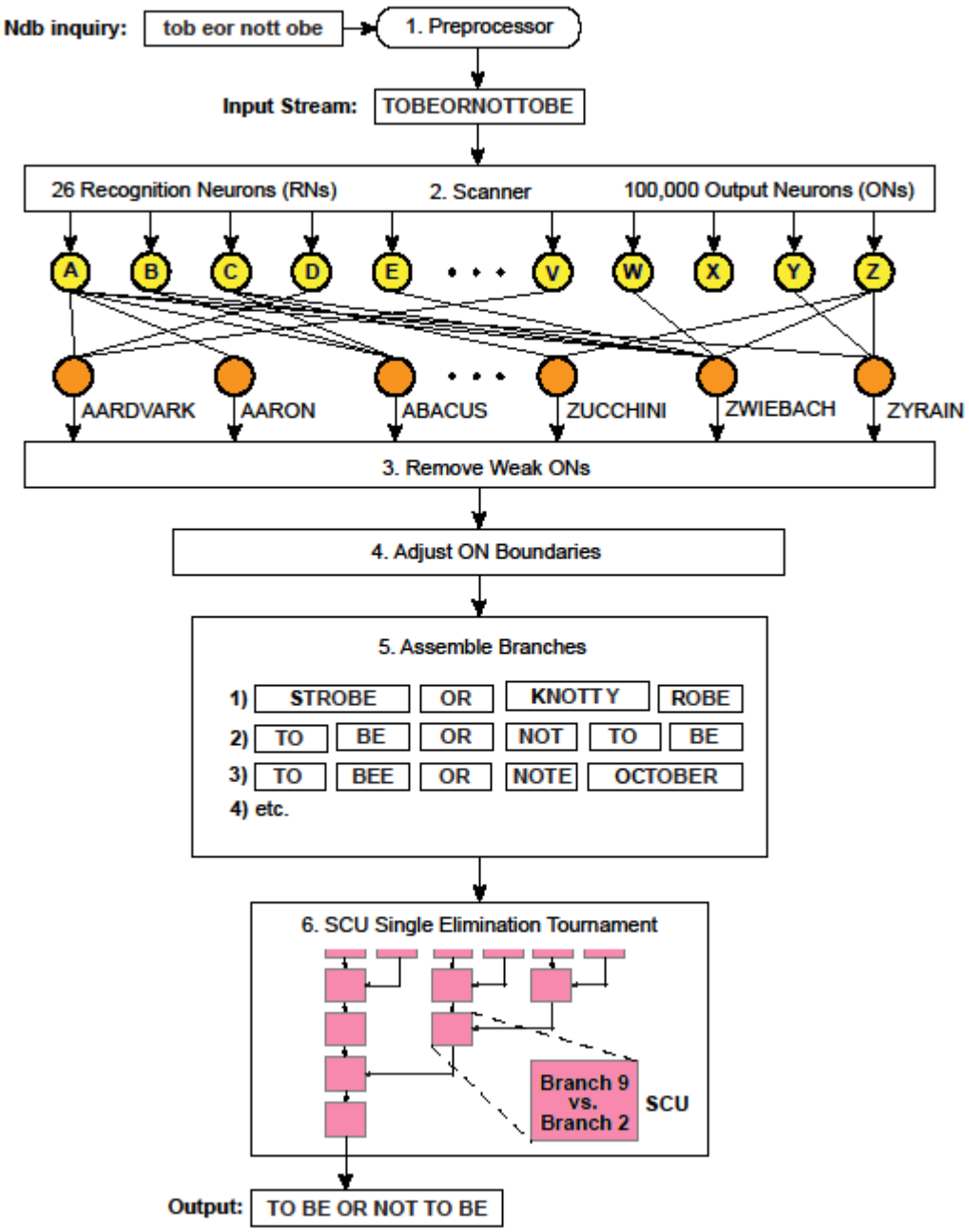

Fig. 2. The principal processing steps in the Neural Database.

The Source of Intelligence

The SCU models the process by which the nervous system in a vertebrate embryo connects to muscles throughout the developing organism (Sanes & Lichtman, 1999). Specifically, axons from two different motor neurons simultaneously establish footholds in the same neuromuscular

junction on the surface of a muscle fiber, as though two electrical plugs were jammed into the same socket. They then compete with each other for control of the junction. An implication of this process is that a competition between two entities may be fundamental to intelligent behavior. It is with this idea in mind that the SCU runs its single elimination tournament.

Competitions in the Simple Competitive Unit

An SCU competition begins by calculating each competitor's Stand-Alone score. Starting with an initial value of zero, every RN in the Input Stream which can be matched with an RN in an ON's Recognition List increases the Stand-Alone score by an increment generated from the SCU's scoring functions (see below). The increment can be enhanced by RNs which are in-order, or it can turn negative, decreasing the Stand-Alone score, for RNs which are out-of-order.

Along with the Stand-Alone score, Table 1 lists four parameters which accompany each ON in the SCU. Table 2 lists seven competitive "Agents" which, based on the data in Table 1, trigger various spike trains. The Agents are listed in the order in which they are activated in the SCU. Their spike trains will modify the Stand-Alone scores of the two contestants and determine the winner of the competition.

Table 1. SCU parameters accompanying each ON competitor.

Parameter	Description
PER	The "Percent Recognition" of an ON, calculated by dividing two SCU Stand-Alone scores: SCU1 / SCU2, where SCU1 is determined from the RNs that the ON claims in the Input Stream and SCU2 is determined from all the RNs listed in the ON's Recognition List.
QUAL	The "Quality" of the recognition, which is simply the absolute value of the difference between the length of the ON's Recognition List and the number of positions that the ON occupies in the Input Stream. Smaller values represent a higher quality of recognition.
cntA	An ON's "Anomaly Count," which is the number of RNs out-of-order or missing from the Input Stream plus the number out-of-order or missing from the Recognition List.
LEN	The number of RNs in the Input Stream which are found in an ON's Recognition List.

Table 2. SCU competitive Agents and the spike trains they trigger.

		Spike Trains	
Agents	Description	Competitor A >	Competitor Z
1. SpaceB	In a branch with multiple ONs, this is the number of boundary crossings supported by a space. If both contenders claim the same space, the one with the higher PER value is counted.	0	-3
2. Anomaly	This is the total cntA+QUAL, summed over all the ONs in the branch.	-4	+2
3. Rec	This is the total Len*(PER-cntA-QUAL), summed over all the ONs in the branch.	+2	-2
4. MinPR	This is the minimum PER among all the ONs in the branch.	0	-6
5. Bound	In a branch with multiple ONs, this is the number of boundary crossings that do not begin with a space. This spike train will only be executed if Anomaly(Z) is less than or equal to Anomaly(A).	-1*(D+1)	+1*(D+1)
6. UnCount	This is the number of RNs unaccounted for in the Input Stream territory claimed by the branch.	-2*D	+3*D
7. MisLead	This is the number of RNs missing from the beginning of each ON in the branch.	-2*D	If D>1, +2

In Table 2, opponent A is whichever contestant has the greater value described in the table. The difference between the A and Z values is represented by D. Positive spike trains (+) are excitatory and will increase the competitor's score while a negative spike train (-) is inhibitory, decreasing a contestant's score. For example, since opponent A has an Anomaly value greater than opponent Z, A's score will receive 4 inhibitory spikes (anomalies are bad) while Z's score will receive 2 excitatory spikes.

The Rec Agent's calculation, Len*(PER-cntA-QUAL), is noteworthy in that the units of measure do not match. PER is a percentage, but both cntA and QUAL are simple counts. This discrepancy is not a concern in the Neural Database since, as a model of an evolved biological system, attending to units of measure would be meaningless in evolution.

In a train of spikes, not all spikes are equal. Each succeeding spike has a diminished impact on the recipient's score. The diminishment is calculated from the SCU's sigmoid scoring functions, which are implemented as:

Excitatory Spike: New_Score = (Current_Score * 0.9011) + 9.89

Inhibitory Spike: New_Score = Current_Score * 0.9011

As shown in Fig. 3, these functions have a fast "excitatory" rise from low values followed by an asymptotic approach to a maximum of 100, and a fast "inhibitory" fall from high values to an asymptotic approach to zero.

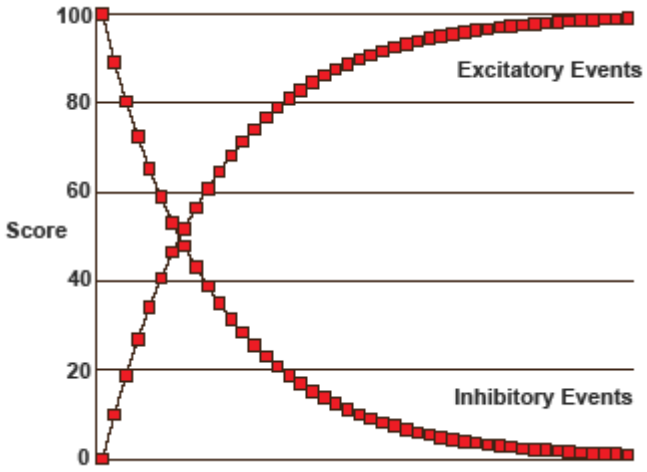

Fig. 3. SCU sigmoid scoring functions.

During an SCU competition, if both contestants achieve the same score, both move up to the next round of competitions, but they are flagged to never compete against each other again. If there is an odd number of contestants, the straggler moves up to the next round. When no more competitions are possible, the tournament ends. Multiple winners can occur. Querying a large word-recognizing Ndb with the text "ens" can result in winners MENS, HENS, LENS, etc.

Modeling Cerebral Processing

Neural Databases can be linked into chains wherein the output of one becomes the input to another. The terminus of the chain can be an Ndb with ONs which execute Actions in response to the initial inquiry. For example, a word-recognizing Ndb (see Fig. 1) can send its output of words to a 'question-recognizing' Ndb (see Fig. 4) in which each RN is a word and each ON is a question with an Action that will provide an answer.

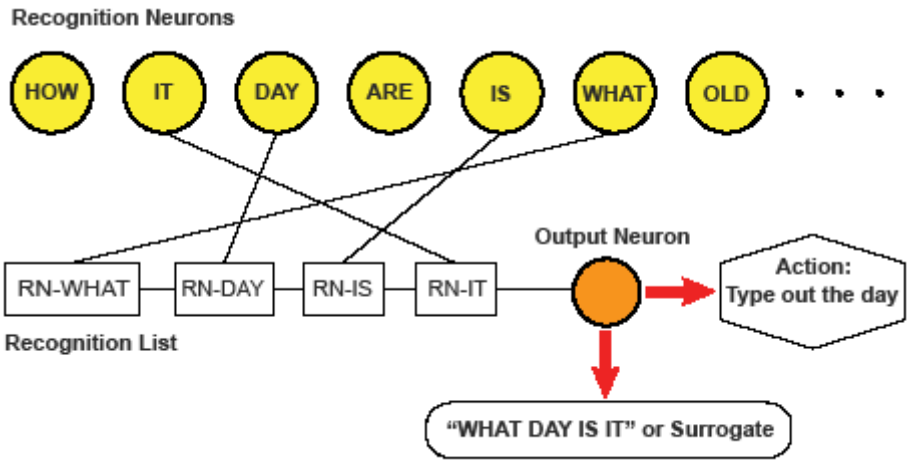

Fig. 4. Data stored in a question-recognizing Neural Database.

As shown in Fig. 5, two such databases, Ndb #1 and Ndb #50, are linked together, responding to different, corrupt inquiries. Both inquiries produce the same result. The second inquiry, "Dayiswht it?," is recognized in Ndb #50 as a word-misspelling of the ON: "WHAT DAY IS IT."

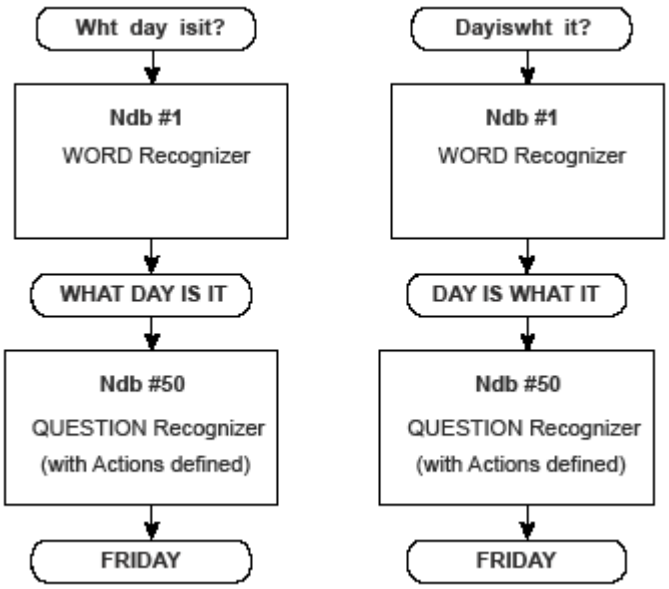

Fig. 5. Two Neural Databases chained together, running two different inquiries.

Teaching verses Training

Neurons in an artificial neural network are effectively cast in stone once their training has concluded. The significant parameters in these networks are the final numeric values assigned to connection weights and neuron biases. The goal of neural network research has consistently been to reduce the errors in these systems – to find the best weights and biases in a myriad of weights and biases. For example, after more than two decades of neural network designs, producing progressively better results in recognizing images of handwritten digits, the error rate dropped from 7.6% (LeCun, Bottou, Bengio & Haffner, 1998) to only 0.09% (An, Lee, Park, Yang & So, 2020). That last effort got only 9 errors out of 10,000 test images.

The human brain would easily deal with the 9 error images by simply adding them to its knowledge. To do the same with a neural network would mean adding the failed images to the 60,000 training images and retraining the network. But this would be an absurdity in the biology of the brain. It would mean that our central nervous system would have to store more than 60,000 original images with no guarantee that the retrained network would be free of errors.

To store in an Ndb the 60,000 images of hand-written digits from the MNIST Training Set (Deng, 2012), each image can be reduced to a Recognition List containing only 10 RNs (see the supplementary information for the procedure). When the 10,000 images from the MNIST Test Set are run through the Ndb, it fails to recognize the correct digit in 161 images, an error rate of 1.61%. Adding this handful of failed test images to the Ndb reduces the failure rate to only 0.03%, a dramatic but predictable improvement. However, updating a Neural Database with more data is a trivial effort, demonstrating this paradigm's duplication of the speed with which the human brain can improve by learning from mistakes.

The 3 new test image errors above can also be added to the Ndb in a fraction of a second, thereby achieving an error rate of 0% with these MNIST datasets.

Models of Incremental Evolution

Incremental evolution via duplicate-then-modify mutations creates biological components with similarities. Such similarities can be seen in many of the Agents in Table 2. The Neural Database itself, in its entirety, can serve as an example of incremental evolution in which the only modification is in the type of data being stored. The recognition of the MNIST images, for example, is accomplished by using hundreds of Ndb's, all of them identical in structure and operation – only the stored data is different (see the supplementary information).

Modeling Improvements in Cognition

Using a library containing numerous test cases (see TestLib.txt in this distribution), word-recognizing Ndb's of various sizes can be used to illustrate the impact of the various spike trains

listed in Table 2. Tables 3 and 4 display 12 Ndb's of increasing size from 86 to more than 96,000 words, each containing the minimum set of words (see WordMin.txt) needed to pass all the test cases in the Test Library.

The St-Alone row in Table 3 lists the number of failed test cases when only the Stand-Alone scores are used to determine the winners of the SCU competitions. In the next row the Stand-Alone scores are altered by adding the SpaceB spike trains. Both the SpaceB and Rec spike trains are used in the following row, and so forth down the column. Note that Table 3 does not include the Anomaly spike trains. However, Table 4 shows the results of repeating all of these tests, this time with the Anomaly Agent included.

Table 3: Test Library errors *WITHOUT* the Anomaly Agent.

Ndb:	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
#words:	86	179	273	461	837	1589	3093	6097	12107	24126	48155	96237
St-Alone	3	3	4	10	10	13	17	22	32	33	40	45
+SpaceB	1	1	2	8	8	10	13	18	23	24	30	34
+Rec				1	1	1	2	2	3	3	4	3
+MinPR	5	5	5	4	3	3	4	5	8	13	15	18
+Bound	5	5	5	4	3	3	4	5	8	14	18	22
+UnCount								1	1	1	1	2
+MisLead								1			1	2

Table 4: Test Library errors *WITH* the Anomaly Agent included.

Ndb:	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
#words:	86	179	273	461	837	1589	3093	6097	12107	24126	48155	96237
St-Alone	3	3	4	10	10	13	17	22	32	33	40	45
+SpaceB	1	1	2	8	8	10	13	18	23	24	30	34
+Anomaly	14	14	14	13	12	12	11	12	16	16	12	14
+Rec	2	2	2	2	1	1	2	3	3	2	1	3
+MinPR	14	14	14	13	12	12	13	14	17	14	13	17
+Bound	14	14	14	13	12	12	13	14	17	18	19	21
+UnCount									1	2	1	1
+MisLead												

The addition of the Anomaly spike trains generates a dramatic increase in errors throughout most of the SCU processing. This highlights the fact that SCU Agents are not simply filters leading gradually to more refined results, but rather competitors which uniquely raise and lower

scores. When combined with the full set of SCU spike trains, the apparently disruptive behavior of the Anomaly spike trains ultimately induces improvements in dealing with large databases.

As seen in Table 2, there are similarities in the Anomaly and Rec Agents which model the products of incremental evolution. Step-by-step improvements, such as those presented in Tables 3 and 4, offer explanations for advancements over time in the ability of human beings to use ever larger vocabularies.

Conclusion

Although this implementation of the Neural Database is inceptive, it nevertheless provides a basic introduction to a paradigm of "built-in intelligence," as opposed to the "trained intelligence" of neural networks. While the Ndb is capable of demonstrating artificial intelligence and solves some of the problems with neural networks, such as slow learning, it does have problems of its own. Its reaction time increases as the size of the database increases and its current solution to text recognition is easily broken.

With further research a system of Ndb's may achieve a complete explanation for a nervous system. However, it should be noted that in a biological Neural Database, every ON would have to be connected to every other ON in order for the SCU to conduct its competitions. Intelligent agents would add more layers of fully interconnected ONs. These structures could provide the raw material for a subsequent evolution of biological neural networks. With Ndb's handling fast learning and the initial recognition of objects from sensory inputs, the representatives of those objects (the Ndb's ONs) could supply the resident data needed to train and retrain neural networks operating deeper in the central nervous system. Given the complexity of the brain, it would not be surprising if an improved neural model emerged from a combination of the two technologies.

References

An, S., Lee, M., Park, S., Yang, H. & So, J. (2020 October). An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition. arXiv:2008.10400v2[cs.CV] doi:https://doi.org/10.48550/arXiv.2008.10400.

Deng, L. (2012 November). The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine*, 29, 6, 141–142. https://doi.org/10.1109/MSP.2012.2211477.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998 November). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86, 11, 2278–2324. doi:10.1109/5.726791. S2CID 14542261.

Sanes, J. R. & Lichtman, J. W. (1999 March). Development of the vertebrate neuromuscular junction. *Annu. Rev. Neurosci*., 22, 389–442. https://doi.org/10.1146/annurev.neuro.22.1.389.