Supplementary Information on the Neural Database (Ndb)

Contents:

A. Ndb Data Types

Three different types of data are processed in the present implementation of a Neural Database:

Data Type	Description
TEXT	Uppercase letters recognizing words
CENTRAL	Words (all in uppercase) recognizing questions
IMAGE_28X28	28 by 28 grayscale pixel arrays recognizing the ten digits: 0 - 9

When a Neural Database is created it is assigned one of these data-types and can only contain data of that type. Translating inquiry data consisting of letters, words, or image arrays into Recognition Neurons (RNs) serves to create an independence from data-type throughout the Ndb. In the most extreme example of converting inquiry data into RNs, an image of type IMAGE_28X28 (784 pixels) is preprocessed into a Recognition List of only ten RNs - see the section below on **Image Recognition of Handwritten Digits**.

There is one exception to this general independence from data-type. An Ndb of type CENTRAL will skip past the Hit_Threshold and the Anomaly_Threshold (described below). A less stringent filtering of ONs seems to work best for this type of data.

B. Neural Database Processing Steps

As diagramed in Fig. 2 in the main Ndb document, the six general processing steps in the Ndb's response to an inquiry are:

Step 1: Preprocessing the Inquiry

1) For data type TEXT:

    a) Internally the Ndb is uppercase - all inquiries will be converted to uppercase.

    b) All characters not represented by an RN in the database are removed, except spaces between the first and last characters in the inquiry. When data is stored in an Ndb, no RN is created for the space character – its special nature is described in e).

    c) All repeating characters are reduced to 2-of-a-kind. For example, TOOOODAY would become TOODAY.

    d) Paired doubles are reduced to a single character followed by the second pair. For example, this text: FFFFRRRRIIIIDDDDAAAAYYY would be reduced by the previous step to FFRRIIDDAAYY which would then be reduced by this step to FRRIDDAYY.

    e) All multiple spaces are reduced to a single space. The location of each single space is then recorded before finally eliminating all spaces. The location of these single spaces can be useful in the SCU as well as in adjustments to the Begin and End Boundaries that define the territory in the Input Stream controlled by each ON.

    f) Finally, all the remaining characters in the inquiry are converted to their RN codes. This list of RNs is referred to as the Input Stream.

2) For data type CENTRAL:

    a) In the chain of Ndb's, the input into this CENTRAL Ndb consists of words separated from each other by single spaces. Words not in uppercase are converted to uppercase.

    b) All spaces are ignored. Words that are not represented by an RN are ignored, otherwise the word's RN replaces the word, creating the Input Stream.

3) For data type IMAGE_28X28:

    a) The procedure converting one of these images into an Input Stream of RNs is described below in the section: **Image Recognition of Handwritten Digits**.

From this point on, the initial inquiry into the Ndb has been transformed into a list of RN codes, i.e. the Input Stream. The only data the Ndb knows anything about are these RNs and the ONs they connect to.

Step 2: Scanning the Input Stream

The Ndb scanner runs through the RNs in the Input Stream (IS) to identify every ON to which they are connected. A list is created of all these ONs and includes the Begin and End Boundaries of each ON's location in the IS, a territory referred to as the ON's "Bound Section." The boundaries are calculated from the position of the RN in the ON's Recognition List.

Suppose, for example, the IS begins with the two letters (or RNs) T and O, both of which can be found in the word (or ON) OCTOBER. When the scanner encounters the T and connects to OCTOBER, it will add the ON to the list of Bound Sections:

ONs	Begin Boundary	End Boundary	RNs
OCTOBER	-1	5	T

Note that, at this point in the processing, zero and negative values are legitimate boundary locations.

When the scanner reads the second RN, the O, it will calculate two Bound Sections for the two appearances of O in OCTOBER. The first position of O creates a Bound Section with Begin and End Boundaries of (2,8). The second O however has the same Begin and End Boundaries as the already existing Bound Section from the T: (-1,5). In this case the RN will simply be added to that Bound Section.

ONs	Begin Boundary	End Boundary	RNs
OCTOBER	-1	5	T,O
OCTOBER	2	8	O

Once this initial list of Bound Sections has been created, it is processed, ON by ON, to see if any of an ON's Bound Sections can be combined to create new Bound Sections, each with even more RNs. For example, suppose the IS contained this string of text: THURSXZXDAY. Two Bound

Sections for the word THURSDAY would initially be created:

ONs	Begin Boundary	End Boundary	RNs
THURSDAY	1	8	T,H,U,R,S
THURSDAY	4	11	D,A,Y

Considering the order in which RNs appear in this ON's Recognition List, a third Bound Section can be created out of the initial Bound Sections above:

ONs	Begin Boundary	End Boundary	RNs
THURSDAY	1	8	T,H,U,R,S
THURSDAY	4	11	D,A,Y
THURSDAY	1	11	T,H,U,R,S,D,A,Y

The new Bound Section is longer than the word itself but clearly represents the word. If the XZX in the IS had instead been FRI, the new, longer Bound Section would still be created, although it would be unlikely to survive a competition with the Bound Section of the ON representing the word FRIDAY.

A lengthy IS inquiring into a large database with many thousands of ONs can generate a list of Bound Sections numbering in the millions. To prevent such an avalanche of potential solutions from wasting time in the SCU, a series of thresholds are next used to pare down the list.
.

<u>Step 3: Removing Weak ONs</u>

In the list of Bound Sections, a well-recognized ON will have to overcome two thresholds:

1) The Hit_Threshold = 61%
   Each Bound Section is scanned to simply count the number of RNs: the hit_count. This threshold removes any Bound Section with a hit_count less than 61% of the number of RNs in the ON's Recognition List.

2) The Anomaly_Threshold = 39%
   This threshold removes ONs if the RNs in the Bound Section are sufficiently out-of-order. In scanning the length of the Bound Section, an anomaly_count will be incremented for each of the following reasons:

   a) The position of an RN in the Input Stream is not consecutive in relation to the previous RN.

b) The position of an RN in the ON's Recognition List is not consecutive in relation to the previous RN.

Once the anomaly_count has been determined, it is used to filter out ONs via the following three steps:

a) While the Ndb scanner initially considers the full extent of "potential" boundaries for an ON's Bound Section, even including Begin Boundaries less than 1, the "real" Begin and End boundaries are defined by the Bound Section's first RN and last RN at their actual locations in the Input Stream. Once these real boundaries have replaced the scanner's potential boundaries, the next steps can proceed.

b) If a group of ONs have the same Begin and End Boundaries, and the same number of RN hits (from the Hit_Threshold above), only those members of the group with the smallest anomaly_count and the shortest Recognition Lists are acceptable and will continue to the next step.

c) Of the remaining ONs, any with an anomaly_count greater than 39% of the length of the ON's Recognition List will be rejected.

These two thresholds will eliminate many of the Bound Sections generated by the Ndb scanner in Step 2 above.

Step 4: Adjusting ON Boundaries

The extent of an ON's territory in the Input Stream, its Bound Section, can be changed by neighboring ONs, including the potential elimination of an ON, either through envelopment by a more powerful ON, or by an extensive reduction of its territory. The "power" of an ON is not a single value. There are several measurements of a Bound Section which can help establish its influence among other ONs.

Along with the Begin and End Boundaries, each ON is assigned the following set of data:

1) ISP: This is a list of the positions recognized in the Input Stream. At each of these positions an RN from the ON's Recognition List was found.

2) RNP: This is a list of the positions recognized in the ON's Recognition List. At each of these positions an RN from the Input Stream was found.

3) PER: This is the "Percent Recognition" of an ON. It is calculated by dividing two SCU Stand-Alone scores: SCU1 / SCU2, where SCU1 is determined from the RNs in the Bound Section and SCU2 is determined from the full set of RNs in the ON's Recognition

List. See the description of the SCU Stand-Alone score below in the section Step 6: Running the SCU Single-Elimination Tournament. There are three pertinent modifications to the calculation of PER:

a) SCU1 will be reduced by the number of extra times an RN in the Recognition List is hit by RNs from the Input Stream. For example, if the Bound Section contained FRIFRIDAY and the ON contained FRIDAY, there would be 3 extra hits.

b) It is possible for SCU1 to be greater than SCU2, in which case PER will be set to 100.

c) If PER=100, it will be reduced to 90 if the length of the Bound Section is greater than the length of the ON's Recognition List. This would be the case described above with FRIFRIDAY.

4) QUAL: This is the "Quality" of the recognition. It is the absolute value of the difference between the length of the ON's Recognition List (the number of letters that spell a word) and the number of positions in the ISP above. Smaller values represent a higher quality of recognition.

5) Composite_Score: To further differentiate the power of one Bound Section in comparison to another, the calculation of this "Composite Score" is designed to exaggerate any differences based on the values above plus the following:

a) LBS: the length of the Bound Section (EndBoundary – BeginBoundary +1)

b) anomalyIS: The number of positions in the ISP that are missing. For example, consider this inquiry:

Inquiry	Input Stream
"th i sdy"	THISDY

In the ON for the word THURSDAY, only these RNs would be recognized: THSDY. For this ON anomalyIS = 1 because the *position* of the "I" in the Input Stream is missing in THURSDAY's list of Input Stream Positions (ISP).

c) anomalyRN: The number of times one or more consecutive positions are missing in RNP. For example, if the ON's Recognition List is FRIDAY and the IS contains FRIDY, the missing "A" would count as an RN anomaly. Anomalies in the Recognition List are not quite as potent as anomalies in the Input Stream (anomalyIS). Each RN anomaly is counted twice, but once the total is determined, it is divided by 3 and only the integer value is retained.

From these elements, the Composite_Score is calculated as follows:

C = PER*PER*LBS*LBS / (1+QUAL)
C = C / (10**anomalyIS)
C = C / (10**anomalyRN)

With this set of data, enveloped weaker ONs can be identified and removed from consideration, plus significant adjustments can be made to the boundaries of weaker ONs that intrude into the Bound Sections of stronger, neighboring ONs. Specifically, the following five boundary reassessments are made:

1) Excluding Enveloped ONs

The Begin to End Boundaries of all the ONs establish physical territories across the Input Stream which can envelop and eliminate some contenders. If the inquiry into the Ndb contains this corrupt text: FRIDY, then both FRIDAY and DAY are viable solutions for that part of the inquiry. But since the Bound Section for FRIDAY (i.e. FRIDY) fully envelopes the Bound Section of DAY (i.e. DY), the shorter word can be removed, leaving the longer word in control of that section of the Input Stream.

The elimination of enveloped ONs begins by iterating through all the ONs from the largest Composite_Score to the smallest. An ON with a high Composite_Score can eliminate a weaker ON if the following four conditions are met:

a) The stronger ON must have a PER that is greater than the Envelopment_Threshold = 75%.

b) The Begin and End Boundaries of the stronger ON must contain or match the Begin and End Boundaries of the weaker ON.

c) The PER of the weaker ON must be less than 100%.

d) The number of RNs in the RNP of the stronger ON must be greater than or equal to the number of RNs in the weaker ON's RNP.

2) Retract Anomalous Boundaries

Anomalous trailing RNs can be removed from an ON, shortening its End boundary, if by doing so the shortened Bound Section leads to a higher Composite_Score. For example,

suppose the following inquiry is entered:

Inquiry	Input Stream
“This is”	THISIS

This input can generate the following Bound Sections, including the ON for THIS with the last two RNs repeated:

ON	Begin Boundary	End Boundary	RNP
THIS	1	6	1,2,3,4,3,4
IS	5	6	1,2

When the two anomalies at the end of THIS, namely the repetition of “3,4”, are removed, the ON will have a higher Composite_Score and the new list of possible interpretations of the inquiry becomes:

ON	Begin Boundary	End Boundary	RNP
THIS	1	4	1,2,3,4
IS	5	6	1,2

An anomalous End Boundary is detected and stripped from an ON’s Bound Section if the following five conditions are met:

a) The ON’s PER must not be 100%.

b) The ON’s RNP is searched from the end toward the beginning to determine if a position is out-of-order. If so, the equivalent position in the ON’s ISP must be a Begin Boundary for some other ON, referred to as ON2. In the above example ON2 is the word IS. Furthermore, ON2 must not be yet another Bound Section of ON, i.e. ignore multiple copies of the same ON that are overlapping.

c) The value of (PER - QUAL) must be less than (PER2 - QUAL2), indicating that ON is less well recognized than ON2.

d) If the End boundary of ON is retracted, will the removed positions in ISP all be found in ISP2? If not, then do not retract ON’s End Boundary.

e) Finally, if the End boundary of ON is retracted, will the ON acquire a higher Composite_Score? If so, then retract its End boundary.

3) Retract Weak Boundaries

Any dominate ON can be used to retract the boundaries of weaker ONs. Dominate ONs are defined as those with:

a) the highest PER among all the ONs, and

b) a preceding space from the list of recorded single spaces (generated by the Preprocessor). Note that the Input Stream does not begin with a space, therefore all ONs with Bound Sections that are at the beginning of the Input Stream are excluded from being dominate ONs.

If any of these dominate ONs overlap the Bound Section of any ON with a lower PER value, then the weaker ON will have its boundaries reduced by giving up the RNs that belong to the dominate ONs. For example, if this is the inquiry:

Inquiry	Input Stream
"Thurs today"	THURSTODAY

then two Bound Sections would initially be created:

ON	Begin Boundary	End Boundary
THURSDAY	1	10
TODAY	6	10

By extending its End Boundary over the RNs: T and O, the ON for THURSDAY ends up claiming DAY just as TODAY does. Note that in the inquiry there is a space between "Thurs" and "today." This, along with its high PER value (100%), makes the ON for TODAY a dominate ON, which allows it to strip DAY from THURSDAY. The resulting Bound Sections become:

ON	Begin Boundary	End Boundary
THURSDAY	1	5
TODAY	6	10

A different result could occur if the space were not there, or if TODAY were corrupted in some way.

Finally, an ON weakened by having RNs stripped from it can be removed entirely if either:

a) its Bound Section has been reduced to only one position, or

b) its new PER is less than the Retract_Boundary_Threshold = 60%.

4) Expanding Boundaries

An ON's Bound Section can be expanded if relevant RN patterns are detected before and/or after its boundaries. Also, merging overlaps by duplicated ONs may lead to an expansion of a Bound Section. Consider, for example, the following:

Inquiry	Input Stream
"Frifrfrifriday"	FRIFRFRIFRIDAY

The recognition of this inquiry will result in the Begin Boundary of the ON for FRIDAY extending to cover the repetitions.

The check for any such expansion begins by first identifying all the positions in the Input Stream which are "owned" by ONs with PER values of 100% *and* QUAL values of zero. These ONs are so well recognized they are effectively "perfect" and their RNs cannot be taken from them by any other ON.

With these "perfect" RNs identified, all the ONs are iterated through from the highest to the lowest PER values. Each ON's Bound Section is checked for the following limitations:

a) If the ON's Begin Boundary is already at the beginning of the Input Stream, obviously no expansion of that boundary is possible - likewise if its End Boundary is at the end of the IS.

b) In the list of interior spaces, if there is a space at the beginning of the ON's Bound Section, no expansion of the Begin Boundary is possible – likewise if the ON's End Boundary has a space right after it.

An ON that makes it past these two conditions needs to check neighboring RNs in the Input Stream for a pattern that matches the beginning or ending of the RNs in its Bound Section.

If such a pattern is detected, it will be ignored if it is either interrupted by a space, or if any of the RNs in the pattern are owned by one of the "perfect" ONs described above. If neither of these blockages are encountered, the pattern will extend either the Begin or End Boundary of the ON. The new size of the Bound Section will cause a recalculation of the ON's various parameters: PER, QUAL, Composite_Score, etc.

5) Post-Boundary Assessment

The boundary adjustments described above may have caused some ONs to become poorly recognized. An ON will be removed from further consideration if the length of its Bound Section times its Composite_Score is less than the Weak_ON_Threshold = 0.035.

Step 5: Assembling Branches

SCU competitors are referred to as "branches" due to the branching structures they emulate when lists of multiple ONs are assembled into potential solutions to the inquiry. A branch is a sequential list of ON Bound Sections wherein each End Boundary is followed by a neighboring Begin Boundary with the complete assembly covering as much of the extent of the Input Stream as possible. For example, in an Ndb with many thousands of words, if the following inquiry were made:

Inquiry	Input Stream
"Wednes today"	WEDNESTODAY

some of the many branches of ON Bound Sections that could represent this Input Stream are:

1) WEDNESDAY TODAY
2) WEDNESDAY TO DAY
3) WED NEST TODAY
4) WED NEST TO DAY
5) etc.

Once all the branches have been assembled, some branches can be eliminated if they contain a single "weak" ON. Whether an ON is weak or not depends on its PER value compared to a dynamic threshold, the value of which depends on the total number of branches:

Number of Branches	Lowest PER Allowed
500 or Less	All branches are allowed.
More then 500	PER >= 80% (PER_1_THRESHOLD)
More than 1,000	PER >= 85% (PER_2_THRESHOLD)
More than 3,000	PER >= 90% (PER_3_THRESHOLD)

For example, if there are 1,234 total branches, then any branch that includes an ON with a PER value below 85% will be removed.

Lastly, the only branches allowed to compete in an SCU tournament are those which meet either of the following two conditions:

1) the branch must be as long as the longest branch, or

2) the branch must be as long as the branch with the highest cumulative Composite_Score: the sum of all the Composite_Scores in the branch.

The final set of branches can then be sent to the SCU to decide which branch is the most likely representation of the initial inquiry.
.

Step 6: Running the SCU Single-Elimination Tournament

The SCU emulates the genesis of a neuromuscular junction (see the main Ndb document) by pairing up branches and running competitions between them. Consider, for example, the process of recognizing the following misspelled word:

Inquiry	Input Stream
"Wednesady"	WEDNESADY

In an Ndb with many thousands of words, at some point during the numerous one-on-one competitions in the SCU, the following two words/ONs could find themselves in contention over the above Input Stream:

UNREADY vs. WEDNESDAY

This situation in the SCU is depicted below in Fig. 1 below. Once the competing branches have been loaded into the SCU and their Stand-Alone scores have been initialized to zero, a scan through the Input Stream (IS) is performed to match the inquiry RNs with RNs in the Recognition Lists of the two competitors.

If an RN appears at multiple locations in a Recognition List, such as the two occurrences of D in the word WEDNESDAY, the location selected for the RN match will be whichever position provides the greatest positive enhancement to the score. The enhancement is a counter which measures the orderliness in which RNs from the Input Stream match up with RNs in the Recognition List. With each succeeding RN match, RNs already matched in order increase the enhancement counter. However, if an RN is out-of-order in comparison to any matched RNs, then the enhancement counter turns negative by a measure of how out-of-order the RN is.

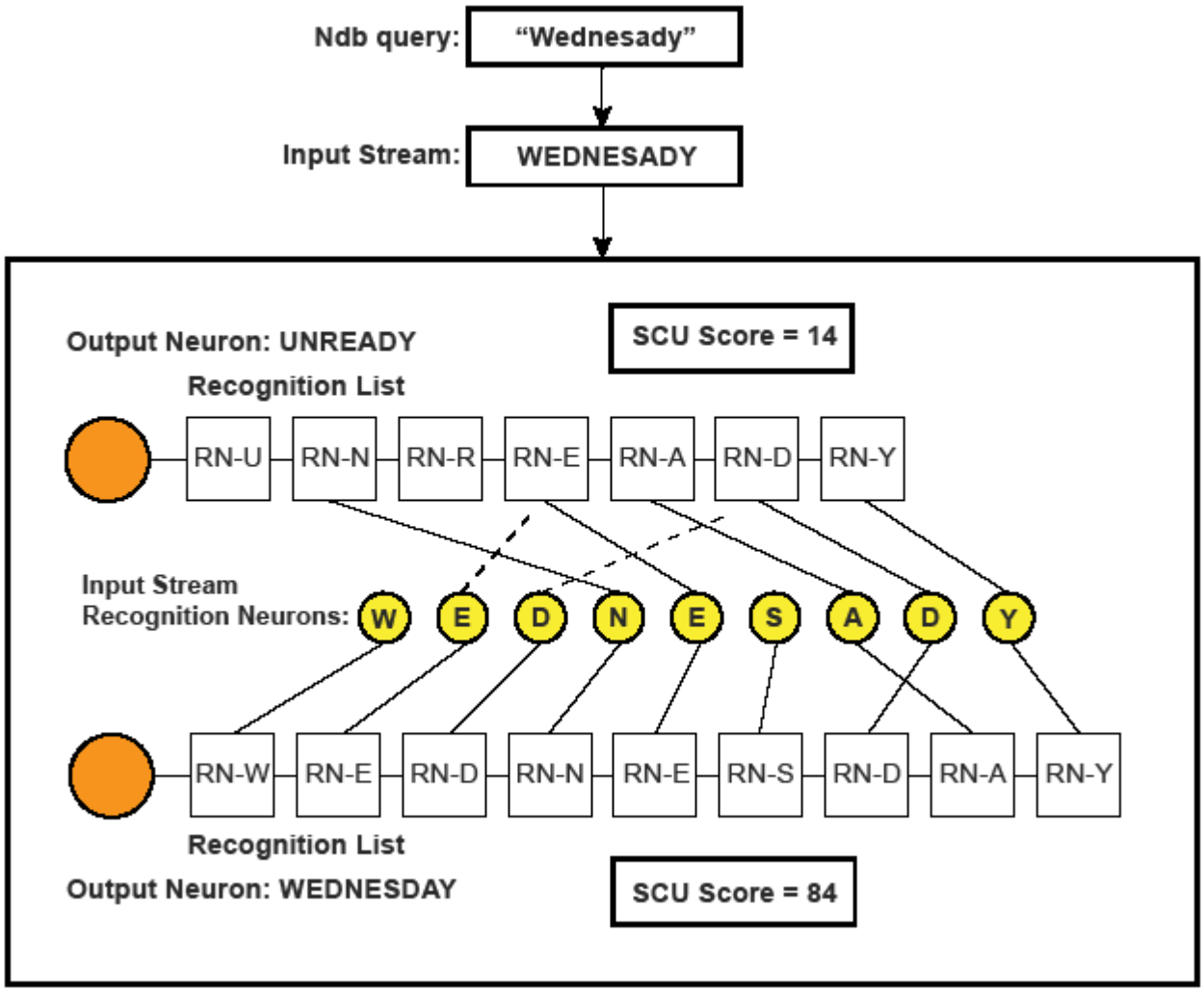

Fig. 1. The Simple Competitive Unit (SCU).

Replacing an already matched RN is also possible. For example, in Fig. 1 the second RN in the Input Stream is the letter E which appears as the fourth letter in the word UNREADY. Initially the SCU will assign that E to that fourth position. However, when the SCU scan reaches the fifth letter in the IS, also an E, it will replace its initial assignment with this "better-E" since it follows, in order, the N from the Input Stream matched with the N in UNREADY. Also in the figure, the third RN in the Input Stream is the letter D which appears as the sixth letter in the word UNREADY. Initially the SCU will assign that D to that sixth position. However, when the SCU scan reaches the eighth letter in the IS, also a D, it will replace its initial assignment with this "better-D" since it follows the A in order.

With each RN match, the enhancement counter determines the number of times the SCU scoring function will be called. See the main Ndb document for a description of the scoring function calculation. If the enhancement counter is positive (en = 0, 1, 2, …), then en+1 excitatory spikes will be generated, increasing the contestant's Stand-Alone score. But if the

enhancement counter is negative (en = -1, -2, …), then -en inhibitory spikes will be triggered, decreasing the Stand-Alone score. Nothing is done to “un-do” or otherwise “back-out” any initial scoring effect that a replaced RN may have previously contributed to the Stand-Alone score.

Lastly, any Input Stream RNs not found in a contender’s Recognition List, such as the W and S not appearing in UNREADY, will be counted and will later trigger the UnCount Agent in the SCU (see Table 2 in the main Ndb document).

Once this preliminary SCU scan of the Input Stream has been done and the Stand-Alone scores for each contestant have been determined, several other essential parameters (PER, QUAL, etc.) are calculated, as described above. With all this in place the competition can proceed. Table 2 in the main Ndb document lists the competition Agents in the order in which they are activated along with the various spike trains they trigger. These in turn will modify the Stand-Alone scores to identify the winner of the competition.

Winners of SCU competitions move to the next round in the tournament. Each new round will generally have half the number of contestants as the previous round. Any contestant that cannot be paired due to an odd number of competitors will automatically move up to the next round. If the SCU generates the same score for both contestants, both move up to the next round but they are marked to never compete against each other again. It is therefore possible to have multiple tournament winners if the final contestants all have identical scores.

C. Image Recognition of Handwritten Digits

Most of this document describes the Neural Database as a system recognizing words in a string of letters. But where did the perfectly recognized letters come from? They would have been produced by a lower-level recognition system capable of converting raw sensory data into identified objects. To demonstrate the Ndb's ability to process sensory data, the following procedure identifies handwritten digits.

The MNIST dataset consists of 70,000 images of handwritten digits divided into two groups: 60,000 images for training a neural network and 10,000 for testing (see the References in the main Ndb document). Each MNIST image is a 28x28 array of pixels. Each pixel has a value in the range from 0 to 255. Instead of storing an actual MNIST image in an Ndb, 399 different "views" of the image are created with each view assigned its own Ndb. To "store an image" in an Ndb, each view is used to generate ten image RNs which form the Recognition List for the ON/digit in that view. The following preprocessing method is used to create the views and generate the image RNs:

1) An image is first reproduced at 7 different contrasts: 2, 33, 66, 100, 133, 166, and 200, where any pixel below the contrast value is set to zero.

2) Each of these 7 images is then simplified with the following translations:

    a) All zeros that are fully surrounded by numeric values greater than zero are replaced with "+" characters. These characters identify enclosed areas.

    b) All other zeros are replaced with a space.

    c) All remaining numeric values are replaced with the "n" character.

3) For each of these altered images, two new views are created to capture different perspectives: a column view and a diagonal view as seen from the upper left corner. In creating the diagonal view, any row shorter than 28 pixels is lengthened to 28 pixels by the addition of zero-valued pixels, i.e. spaces. This process generates an image that is 28 columns by 55 rows. The diagonal view is then reduced to 28x28 pixels by removing all even-numbered rows. See Tables 1 through 4 below for examples of this transformation process on a MNIST image.

4) Finally, from each of these different views, two sets of smaller, overlapping "sub-views" are created, referred to as panels. There are 9 panels at 16x16 pixels that cover the entire view, plus another 9 panels at 18x18 pixels, also covering the entire view.

Once all 399 views have been rendered from the original MNIST image, a Recognition List of 10 image RNs is generated for each of these views. These RNs represent general features of the digit, or partial digit, in the view:

RN1 = CS: Is the digit generally curvy or straight?

RN2 = INT: The number of interior areas completely surrounded by “n” pixels.

RN3 = SLANT: Is the digit slanted left or right or not at all?

RN4 = GIRTH: Is the digit slim, fat, or in-between?

RN5 = LR: Is the longest row of pixels in the top, middle, or bottom of the digit?

RN6 = RWTB: Is the digit top heavy, bottom heavy, or neither?

RN7 = RWRL: Is the digit weighted to the left, right, or neither?

RN8 = CAV: In the digit’s left and right boundaries, where are cavities located?

RN9 = PED: Is the lower half of the digit top-heavy or bottom-heavy?

RM10 = LL: Is the longest line in the digit in a row, a column, or a diagonal?

Each of the 399 Recognition Lists will then be stored in the appropriate Ndb, identifying the ON, i.e. the digit assigned to the MNIST image.

To recognize a MNIST Test image, the image is first run through the above process to produce the 399 sets of image RNs. Each set is an Input Stream which is then used to inquire into the appropriate Ndb. All 399 results are then checked to identify which digit is recognized the most.

It is possible for the same set of 10 image RNs to identify different digits. For example, a 16x16 pixel panel covering the upper left corner of a view may contain nothing but pixels of value zero. Multiple MNIST images of different digits may have the same “blank” panel. The Ndb will store in the ON’s Surrogate the multiple digits identified by any such identical Recognition Lists. If a test image has the same “blank” panel, the Ndb for that panel will return the multiple results, but only if the total number of digits in the group is less than the Image_Ambiguity_Threshold = 8. The fact that some views and some panels can identify more than one digit is not a significant problem since the Neural Database image recognition solution is to use hundreds of Ndb’s to help differentiate images.

It is significant to note that none of the Ndb’s internal thresholds were altered to achieve the

recognition of these handwritten digits. No changes were made to the Simple Competitive Unit. No special code was added or removed except for:

1) the preprocessing used to generate the 399 different views and each view's set of image RNs, and

2) a simple postprocessing count to select the most recognized digit.

Table 1

Pixel Array: Raw MNIST Training Image #47, Contrast=33, rows 1-28, cols 7-22

							152	203	181	141	58				
							40	172	247	188	232	234	35		
					82	101	143	252	245	67	35	225	214		
			132	237	254	254	254	254	254	243	80	210	248	35	
		163	251	211	107		36	120	240	246	98	218	143		
	47	251	166							43	189	212			
	40	250	214							148	250	99			
		137	254	234	103				154	225	85				
			44	195	254	184		129	235	35					
					101	240	254	254	66						
						172	254	254	108						
					154	253	98	190	254	104					
				91	254	131			212	225	64				
				238	254				55	244	195				
			90	254	219					100	254	111			
			127	254	116						248	126			
			127	254	63					49	252	126			
			124	248					38	167	254	101			
				233	211	115	115	135	254	244	130				
				70	236	254	254	254	173	38					

Table 2

Pixel Array: Ndb Row View, Training Image #47, Contrast=33, rows 1-28, cols 1-28

												n	n	n	n	n									
												n	n	n	n	n	n	n							
										n	n	n	n	n	n	n	n	n							
									n	n	n	n	n	n	n	n	n	n	n						
								n	n	n	n	+	n	n	n	n	n	n							
							n	n	n	+	+	+	+	+	+	n	n	n							
							n	n	n	+	+	+	+	+	+	n	n	n							
								n	n	n	n	+	+	+	n	n	n								
									n	n	n	n	+	n	n	n									
											n	n	n	n	n										
												n	n	n	n										
											n	n	n	n	n	n									
										n	n	n	+	+	n	n	n								
										n	n	+	+	+	n	n	n								
									n	n	n	+	+	+	+	n	n	n							
									n	n	n	+	+	+	+	+	n	n							
									n	n	n	+	+	+	+	n	n	n							
									n	n	+	+	+	+	n	n	n	n							
										n	n	n	n	n	n	n	n								
										n	n	n	n	n	n	n									

Table 3

Pixel Array: Ndb Column View, Training Image #47, Contrast=33, rows 1-28, cols 1-28

										n	n															
									n	n	n	n														
								n	n	n	n	n						n	n	n	n					
								n	n	+	+	n	n				n	n	n	n	n	n	n	n		
							n	n	n	+	+	n	n	n		n	n	n	n	n	n	+	n	n		
							n	n	+	+	+	+	n	n	n	n	n	+	+	+	+	+	n	n		
					n	n	n	n	n	+	+	+	+	n	n	n	+	+	+	+	+	+	n	n		
					n	n	n	n	n	+	+	+	n	n	n	n	+	+	+	+	+	+	n	n		
					n	n	n	n	n	+	+	n	n	n	n	n	n	+	+	+	n	n	n		
					n	n	n	n	n	n	n	n	n			n	n	n	n	+	n	n	n	n		
					n	n	n	n	n	n	n	n					n	n	n	n	n	n	n			
						n	n	n	n	n	n								n	n	n	n				
						n	n	n	n																	
								n																		

Table 4

Pixel Array: Ndb Diagonal View, Training Image #47, Contrast=33, rows 1-28, cols 1-28

						n																					
					n	n	n	n																			
					n	n	n	n	n																		
						n	n	n	n	n	n																
							n	n	n	+	n	n															
								n	+	+	+	n	n														
								n	n	+	+	+	n	n													
								n	n	+	n	n	n	n	n	n	n										
							n	n	n	n	n	n	n	+	n	n	n	n									
											n	n	+	+	+	n	n										
										n	n	+	+	+	n	n											
										n	n	+	+	n	n												
									n	n	+	n	n														
									n	n	n																

D. Technical Specifications of the Image RNs

The following parameters are used in some of the calculations described below:

1) The WIDTH of the digit is determined by scanning the view column by column to find the columns of both the first and last digit pixels encountered. The WIDTH is then the distance between the two.

2) The HEIGHT of the digit is determined by scanning the view row by row to find the rows of both the first and last digit pixels encountered. The height is then the distance between the two.

RN1 = CS: Is the digit generally curvy or straight?

The digit is scanned around its boundary. A counter is incremented whenever the next pixel on the boundary is in a different direction from the current pixel. After the scan:

CS = 0, cannot be determined (blank view?)
CS = 10, if the counter is less than CS_THRESHOLD, the digit is straight,
CS = 20, otherwise it's curvy.

where CS_THRESHOLD = 75

RN2 = INT: The number and location of fully enclosed regions in the digit.

Enclosed regions are marked with "+" characters. The view is scanned row by row to find where these characters begin and end. Any of the following five values can be returned:

INT = 0, no enclosed regions were found
INT = 10, there is one enclosed region, but its location is not specified
INT = 20, there is one enclosed region in the upper half of the digit
INT = 30, there is one enclosed region in the lower half of the digit
INT = 40, there are two enclosed regions

RN3 = SLANT: Is the digit slanted left or right or neither?

The view is scanned row by row to find the first row and the last row of the digit. C1 is the column of the first pixel in the first row, and C2 is the column of the first pixel in the last row. The difference between the two is: D = C1-C2. The digit slants to the right if D is greater than zero. It slants to the left if D is negative. The SLANT of the digit is given one of the following values:

SLANT = 0, cannot be determined
SLANT = 10, slanted to the right if D/WIDTH is greater than SLANT_THRESHOLD

SLANT = 20, slanted to the left if -D/WIDTH is greater than SLANT_THRESHOLD

where SLANT_THRESHOLD = 0.1

RN4 = GIRTH: Is the digit slim, fat, in-between, or indeterminate?

The GIRTH of the digit is given one of the following values:

GIRTH = 0, cannot be determined
GIRTH = 10, the ratio: WIDTH/HEIGHT is less than the SLIM_THRESHOLD
GIRTH = 20, in between
GIRTH = 30, the ratio: WIDTH/HEIGHT is greater than the FAT_THRESHOLD

where SLIM_THRESHOLD = 0.4
and FAT_THRESHOLD = 0.8

RN5 = LR: Is the longest row of pixels in the top, middle, or bottom of the digit?

During the row scan, a count is made of the number of pixels in each row that are either a "n" or a "+" character. The first such row encountered is FirstR. The row with the greatest count is LongR. The relative location of the longest row is:

R = (LongR-FirstR+1)/HEIGHT

The LR of the digit is given one of the following values:

LR = 0, cannot be determined
LR = 10, if R < LOWER_THRESHOLD, the LR is in the upper third of the digit
LR = 20, if R < UPPER_THRESHOLD, the LR is in the middle of the digit
LR = 30, otherwise the LR is in the bottom third of the digit

where LOWER_THRESHOLD = 0.34
and UPPER_THRESHOLD = 0.67

RN6 = RWTB: Is the digit top heavy, bottom heavy, or neither?

The view is scanned row by row while counting:

1) Top = total number of "n" and "+" characters in the top third of the view
2) Bottom = total number in the bottom third

The RWTB of the digit is set to one of the following values:

RWTB = 0, cannot be determined
RWTB = 10, if neither of the following…

RWTB = 20, if Top/(Top+Bottom) is less than LOWER_THRESHOLD
RWTB = 30, if Top/(Top+Bottom) is greater than UPPER_THRESHOLD

where LOWER_THRESHOLD = 0.375
and UPPER_THRESHOLD = 0.625

RN7 = RWRL: Is the digit weighted to the left, right, or neither?

The view is scanned column by column while counting:

1) Right = total number of "n" and "+" characters in the right third of the view
2) Left = total number in the left third

The RWRL of the digit is set to one of the following values:

RWRL = 0, cannot be determined
RWRL = 10, if neither of the following…
RWRL = 20, if Left/(Left+Right) is less than LOWER_THRESHOLD
RWRL = 30, if Left/(Left+Right) is greater than UPPER_THRESHOLD

where LOWER_THRESHOLD = 0.375
and UPPER_THRESHOLD = 0.625

RN8 = CAV: How many cavities are there in the digit's boundary and where are they located?

The left side and the right side boundaries of the digit are scanned row by row to see if changes in the column locations of the boundary reveal entering and then exiting a cavity.

The CAV value for the digit is set to one of the following:

CAV = 0, no cavities detected
CAV = 10, one cavity facing left
CAV = 20, two facing left
CAV = 30, one facing right
CAV = 40, two facing right
CAV = 50, one facing left, one facing right
CAV = 60, two facing left, one facing right
CAV = 70, one facing left, two facing right
CAV = 80, two facing left, two facing right

RN9 = PED: Is the lower half of the digit top-heavy or bottom-heavy?

The bottom half of the view is scanned row by row while counting:

1) Top = total number of "n" characters in the top half of the view bottom
2) Bottom = total number in the bottom half of the view bottom

Only “n” characters are counted since an interior space will not contribute to the detection of a possible pedestal or a mid-view bulge.

PED is then set to one of the following values:

PED = 0, cannot be determined
PED = 10, if (Top+Top) is less than Bottom, the lower half of the digit is bottom-heavy
PED = 20, if (Bottom+Bottom) is less than Top, the lower half is top-heavy

RM10 = LL: Is the longest line in the digit in a row, a column, or a diagonal?

The line lengths from the various perspectives count only the occurrence of consecutive “n” characters. The LL value for the digit is set to one of the following:

LL = 0, cannot be determined
LL = 10, the longest line is in a digit row
LL = 20, in a column
LL = 30, in a diagonal as seen from the upper left of the view
LL = 40, in a diagonal as seen from the lower left

Finally, as described above in Step 2: Scanning the Input Stream, the Ndb scanner creates Bound Sections by matching inquiry RNs from the Input Stream with RNs in an ON’s Recognition List. If there are multiple positions of an RN in a Recognition List, such as the S in MISSISSIPPI, then multiple Bound Sections will be generated to help deal with varieties of misspellings. Because of this however, if two different image RNs in the same Recognition List have the same numeric value, the Ndb scanner will automatically create multiple Bound Sections. This creates many more potential solutions for the SCU to resolve.

But the image RNs are, in fact, all unique. Every view has a Recognition List of exactly ten RNs and each RN represents a specific feature of a digit. An image RN with a value of 10 is different from another RN with the same value in the same Recognition List. To prevent unnecessary Bound Sections from being generated, an “offset” is added to each image RN and it is increased by 100 for every successive RN added to the Recognition List. For example, if these are the first 4 image RNs in some Recognition List:

10, 30, 10, 10, …

the offset will alter these values and return this Recognition List:

110, 230, 310, 410, …

The addition of the offset does not make the Ndb any more or less accurate, it just prevents unnecessary processing.

E. Glossary

Term	Description
Action	An ON's executable code
Agent	A set of spike trains and the conditions which trigger them
Begin Boundary	The first position in an ON's Bound Section
Bound Section	Continuous positions in the Input Stream claimed by an ON
Branch	A list of ONs representing a possible interpretation of the inquiry
End Boundary	The last position in an ON's Bound Section
Inquiry	Initial input to the Neural Database
Input Stream	The list of RNs representing the inquiry
IS	Input Stream
MNIST	Datasets of handwritten digits (referenced in the main Ndb document)
Ndb	Neural Database
ON	Output Neuron
Panel	A sub-image (16x16 or 18x18 pixels) of a MNIST image
Query	Initial input to the Neural Database
Recognition List	Ordered list of RNs used to recognize/identify an ON
RN	Recognition Neuron
SCU	Simple Competitive Unit
Spike train	A set of excitatory and/or inhibitory scoring events
Stand-Alone score	An ON's initial non-competitive SCU score
Surrogate	Alternate output for an ON
View	An Ndb rendition/reduction of a raw MNIST image