

Chapter 5

Binary CNN Characterization via Boolean Functions

5.1. Binary and universal CNN truth table

Our objective in this section is to show that every space-invariant *binary* (black-and-white) *CNN* belonging to the *uncoupled* class $\mathcal{C}(A^0, B, z)$ with a 3×3 neighborhood ($r=1$) which maps any static *binary* 3×3 input pattern U into a static binary 3×3 output pattern $Y(\infty)$ can be *uniquely* defined by a *Boolean function* \mathcal{C} of 9 binary input variables¹

$$\mathbf{u}_{ij} = [u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9]^T \quad (5.1)$$

where $u_i \in \{0, 1\}$ denotes one of the 9 pixels within the sphere of influence of cell C_{ij} as shown in Fig.1(a). Note that we have opted for a “single” rather than a “double” subscript notation to avoid clutter. Note also that \mathbf{u}_{ij} has a *subscript* (ij) and is set in a *bold face* type in order to distinguish it from the input u_{ij} (set in light-face type) of cell C_{ij} . Although we can code the 9 pixels u_{kl} , $kl \in \{-1, 0, 1\}$ by any combination of u_i , we have chosen the coding scheme shown in Fig.1(b) for pedagogical reasons that will be obvious later. A simple mnemonic to reconstruct this code is to remember the subscript “5” always refers to the input u_{00} , corresponding to the *center* cell C_{ij} , whereas the subscripts $\{1, 2, 3, 4\}$ refer to the surround cells in the N→E→S→W clockwise compass directions, and the remaining subscripts $\{6, 7, 8, 9\}$ refer to the surround cells in the NE→NW→SE→SW clockwise compass directions.

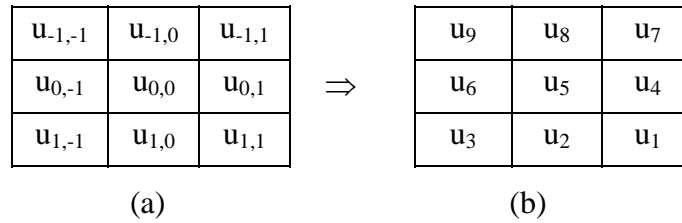


Fig.1 Every 3×3 binary pattern from (a) will henceforth be coded by the standard scheme in (b).

Now given any static binary input pattern U , the color (black or white, since the CNN is *assumed* to be binary) of *any* output pixel is determined uniquely by only a small part of U exposed to a 3×3 transparent window centered at cell C_{ij} , because the sphere of influence $S_1(i,j)$ is *assumed* to be a 3×3 neighborhood. Hence the color $\{0, 1\}$ of the output pixel $y_{ij}(\infty)$ is uniquely determined by the binary value (0 or 1) of the 9 pixels u_i ,

¹ We have chosen here $\{0,1\}$ instead of $\{-1,1\}$ as our binary codes in order to exploit directly the immense literature and theory on Boolean functions, which are almost always couched in terms of “zeros” and “ones”.

u_2, \dots, u_9 exposed by the 3x3 window. This unique answer is obtained by solving the system of MxN ODE having the *prescribed* CNN templates (A^0, B, z), and prescribed initial state $\mathbf{x}(0)$. Now even though there are *infinitely many* distinct templates (recall the coefficients of A^0, B , and z can be any *real* number, which is uncountable), there will be only a finite (albeit very large) number of distinct combinations of 3x3 “checkerboard” patterns of black and white cells, namely, $2^9 = 512$.

Figure 1c shows how a single binary input is represented.

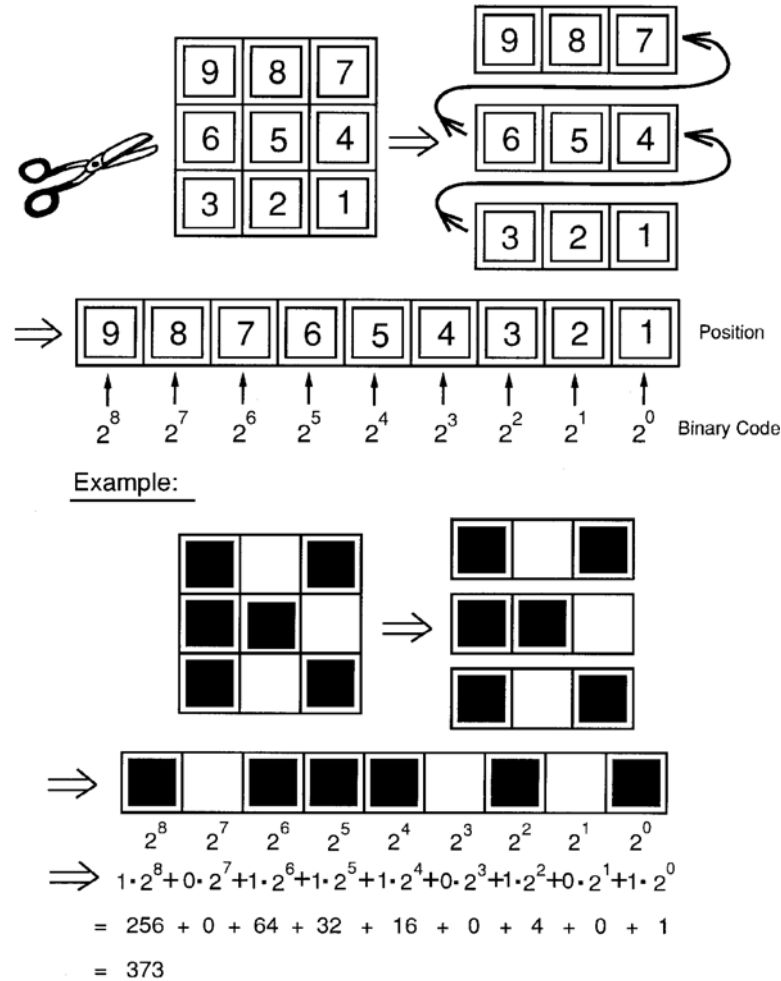


Fig. 1c Representing a single binary input

Since each such pattern can map to either a “0”, or a “1”, there are exactly²

² In order to appreciate how large the number Ω is, compare it to the following universal benchmarks:

- Age of the universe = 10^{30} picoseconds.
- Mass of the universe (calculated in units of mass of a hydrogen atom) = 10^{80}
- Volume of the universe (calculated as a sphere with a diameter of 10 thousand million light-years) = 10^{84} cm^3

$$\Omega \triangleq 2^{2^9} = 2^{512} \approx 1.3408 \times 10^{154} > 10^{154} \quad (5.2)$$

distinct Boolean maps of 9 binary variables. This maps can be ordered in a table shown in Fig. 1d. Each row shows a different binary 9 input 1 output map.

CNN Program NAME	CNN Program Code													
	512	511	510	509	508	507	506	7	6	5	4	3	2	1
Fill White	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Game of Life	0	0	0	1	0	1	0	0	0	0	0	0	0	1
Edge Detector	1	0	1	1	1	1	1	0	0	0	0	0	0	0
2^{512-3}	1	1	1	1	1	1	1	1	1	1	1	0	1	1
2^{512-2}	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Fill Black	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Total Number of Distinct CNN Programs = $2^{512} \approx 1.34078 \times 10^{154}$

Fig. 1d CNN Program Code of 9 variable binary input

Let \mathcal{C}_Ω denote the universe of *all* such maps. Now since \mathcal{C}_Ω is the maximal set, by definition, the Boolean map generated by each member of the standard CNN universe $\mathcal{C}(A^0, B, z)$ must be a member of \mathcal{C}_Ω ³. Hence

$$\mathcal{C}(A^0, B, z) \subset \mathcal{C}_\Omega \quad (5.3)$$

We have just proved the following fundamental result:

Theorem 1 Binary CNN Truth Table

Every binary standard CNN with template (A^0, B, z) and prescribed initial state $\mathbf{x}(0)$ is a member of the universe \mathcal{C}_Ω of all Boolean functions of 9 variables and is therefore uniquely characterized by the *CNN truth table* shown in Fig.2, consisting of 512 *rows* (one for each distinct 3x3 checkerboard pattern), 9 *input columns* (one for each binary input variables u_i), and 1 *output* column whose value (0 or 1) corresponds to $y_{ij}(\infty)$.

binary pattern number	input variables									output $y_{ij}(\infty)$
	u_9	u_8	u_7	u_6	u_5	u_4	u_3	u_2	u_1	
0										
1										
.										
.										
.										
510										
511										

Fig. 2 Truth table for defining *any* Boolean functions of 9 variables.

Theorem 1 gives us *the most rigorous* method for characterizing a space-invariant binary CNN, and is therefore of fundamental importance. Since this table will in general exceed the length of a typical page, let us divide it into 16 component truth tables each one containing 32 rows. For example, the 16 component truth tables which characterize the *Edge* templates are given in Fig.3(a)-3(p). To clarify our notations, in the first component table shown in Fig.3(a), each entry for the input variables is coded by a “0” or a “1”, instead of our earlier notation of “-1” and “1”, in order for us to exploit the extensive theory and literature on Boolean functions, which are almost universally couched in “zeros” and “ones”. Observe that we have ordered the binary values in the truth table in the same order for enumerating the *binary number* 0, 1, 2, 3, ..., 511, consecutively. Since it is usually more pleasing for the eyes to decode a table of black-

³ Note that $\mathcal{C}(A^0, B, z)$ may generate non-Boolean maps as well.

and-white cells than a table of “zeros” and “ones”, we will henceforth code our CNN truth tables by black-and-white cells.

To construct the truth table for any binary CNN $\mathcal{C}(A^0, B, z)$ with prescribed initial state $\mathbf{x}(0)$, simply solve the associate system of differential equations for each input of 512 distinct binary patterns listed in Fig. 2 and fill in the corresponding calculated output, either black (1) or (0). Since the 512 binary patterns are fixed, each corresponding to a 9-bit binary expression of an integer N , $N = 0, 1, 2, \dots, 511$, it is easy to write a computer program to generate the truth table automatically, given any templates (A^0, B, z) and the prescribed initial condition $\mathbf{x}(0)$. In particular, simply assume a 3x3 CNN array ($M=N=3$) and find the solution of the *center cell* C_{00} .

Example 1 Edge CNN

1

	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	1	1	0
4	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	1	0	1	0
6	0	0	0	0	0	0	1	1	0	0
7	0	0	0	0	0	0	1	1	1	0
8	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	1	0	0	1	0
10	0	0	0	0	0	1	0	1	0	0
11	0	0	0	0	0	1	0	1	1	0
12	0	0	0	0	0	1	1	0	0	0
13	0	0	0	0	0	1	1	0	1	0
14	0	0	0	0	0	1	1	1	0	0
15	0	0	0	0	0	1	1	1	1	0
16	0	0	0	0	1	0	0	0	0	1
17	0	0	0	0	1	0	0	0	1	1
18	0	0	0	0	1	0	0	1	0	1
19	0	0	0	0	1	0	0	1	1	1
20	0	0	0	0	1	0	1	0	0	1
21	0	0	0	0	1	0	1	0	1	1
22	0	0	0	0	1	0	1	1	0	1
23	0	0	0	0	1	0	1	1	1	1
24	0	0	0	0	1	1	0	0	0	1
25	0	0	0	0	1	1	0	0	1	1
26	0	0	0	0	1	1	0	1	0	1
27	0	0	0	0	1	1	0	1	1	1
28	0	0	0	0	1	1	1	0	0	1
29	0	0	0	0	1	1	1	0	1	1
30	0	0	0	0	1	1	1	1	0	1
31	0	0	0	0	1	1	1	1	1	1

3

	9	8	7	6	5	4	3	2	1
64									
65									
66									
67									
68									
69									
70									
71									
72									
73									
74									
75									
76									
77									
78									
79									
80									
81									
82									
83									
84									
85									
86									
87									
88									
89									
90									
91									
92									
93									
94									
95									

5

	9	8	7	6	5	4	3	2	1
128									
129									
130									
131									
132									
133									
134									
135									
136									
137									
138									
139									
140									
141									
142									
143									
144									
145									
146									
147									
148									
149									
150									
151									
152									
153									
154									
155									
156									
157									
158									
159									

7

	9	8	7	6	5	4	3	2	1
192									
193									
194									
195									
196									
197									
198									
199									
200									
201									
202									
203									
204									
205									
206									
207									
208									
209									
210									
211									
212									
213									
214									
215									
216									
217									
218									
219									
220									
221									
222									
223									

2

	9	8	7	6	5	4	3	2	1
32									
33									
34									
35									
36									
37									
38									
39									
40									
41									
42									
43									
44									
45									
46									
47									
48									
49									
50									
51									
52									
53									
54									
55									
56									
57									
58									
59									
60									
61									
62									
63									

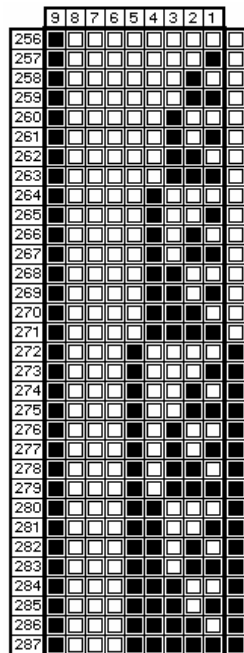
4

	9	8	7	6	5	4	3	2	1
96									
97									
98									
99									
100									
101									
102									
103									
104									
105									
106									
107									
108									
109									
110									
111									
112									
113									
114									
115									
116									
117									
118									
119									
120									
121									
122									
123									
124									
125									
126									
127									

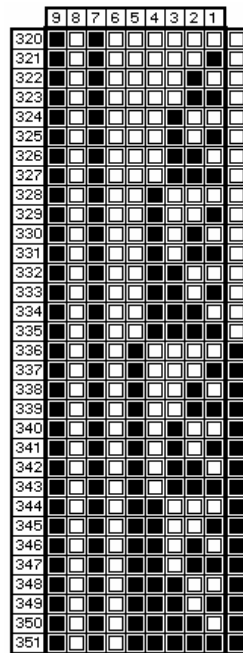
6

	9	8	7	6	5	4	3	2	1
160									
161									
162									
163									
164									
165									
166									
167									
168									
169									
170									
171									
172									
173									
174									
175									
176									
177									
178									
179									
180									
181									
182									
183									
184									
185									
186									
187									
188									
189									
190									
191									

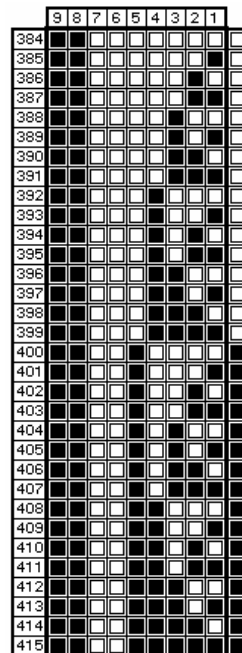
9



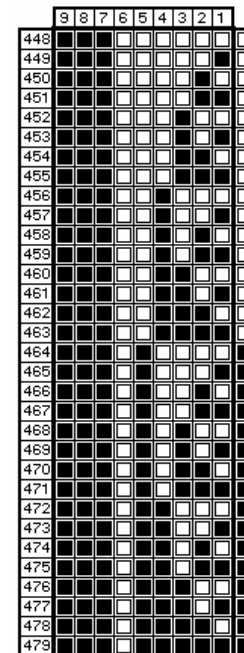
11



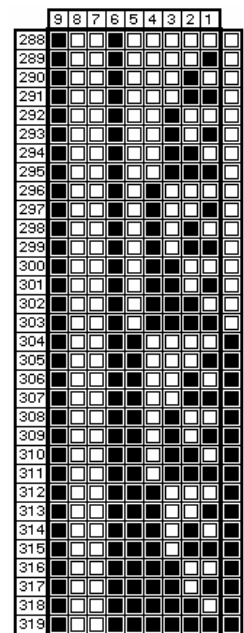
13



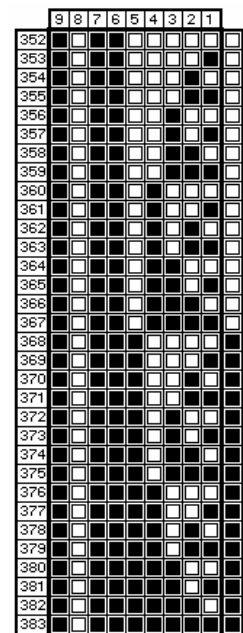
15



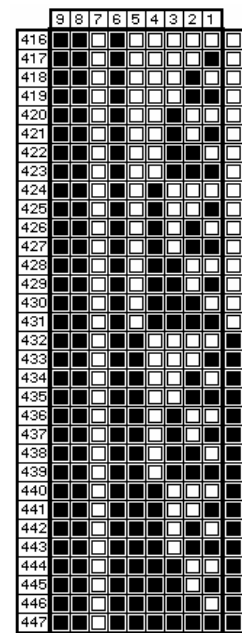
10



12



14



16

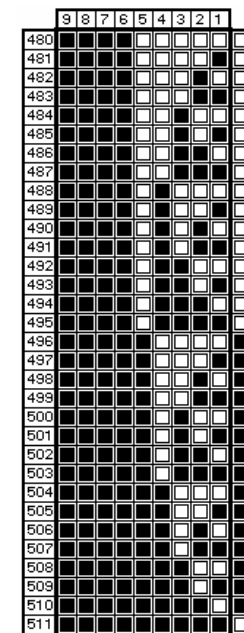


Fig 3(a)-3(p)

The truth table for the edge CNN calculated by the above procedure is shown in Fig.3, decomposed into 16 components, clearly, except for displaying a few of these truth tables for analysis and pedagogical purposes, it is impractical to list the truth table of all useful CNN's. They can, however, easily be stored on a diskette, to be retrieved only when needed. Displaying the truth table on a computer screen has the advantage of showing a continuous table when any part of the table can be scrolled into the entire view.

The alert reader will have already realized that the truth table format of Fig.1 contains a great deal of redundancy. Indeed, in each of the 16 components shown in Fig.3, the domain of the binary input variables u_1, u_2, \dots, u_9 , which constitutes the bulk of the space of each table, remains unchanged. Hence, we only need to record *the last column* of each of these 16 component tables. Since each column has 32 cells, we need only store $16 \times 32 = 512$ pixel values (0 or 1) for each binary CNN $\mathcal{C}(A^0, B, z)$ with prescribed initial conditions and will be able to reconstruct these 16 component tables. For maximum space efficiency, we can pack all 16 columns from Fig.2, each with 32 entries, into 16 rows, next to each other to form a grid containing exactly $16 \times 32 = 512$ cells, as shown in Fig.4. Since this table contains the same information as those of Fig.2, we have achieved an immense amount of *data compression*. Indeed, since this table contains only 512 entries, one for each input pattern, it is a *minimal representation*. We will henceforth refer to Fig. 4 as *minimal CNN truth table*.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1																																
2																																
3																																
4																																
5																																
6																																
7																																
8																																
9																																
10																																
11																																
12																																
13																																
14																																
15																																
16																																

Fig. 4 Minimal CNN truth table

Corollary to Theorem 1

Every space-invariant binary CNN with a 3×3 neighborhood and specified by templates (A^0, B, z) and a prescribed initial state $\mathbf{X}(0)$ is associated with a unique minimal CNN truth table.

Remarks

1. The *uniqueness* assertion in the above corollary is with respect to a given template (A^0, B, z) and initial state $\mathbf{X}(0)$. It is not unique with respect to a given “global task” since a given task in general can be implemented by many distinct CNN templates (infinitely many indeed).

2. The above corollary only asserts that for every CNN template (A, B, z) and initial state, there corresponds, a minimal truth table, or equivalently, a Boolean function of 9 variables. However, the converse is not true, i.e., given a Boolean function $\mathcal{B} \in \mathcal{C}_\Omega$, or its associated minimal truth table, there may *not* exist a CNN template and an initial state $\mathbf{X}(0)$ which yields this truth table. However, we will prove later that every member of \mathcal{C}_Ω , i.e., every Boolean function of 9 variables, can be realized by a *CNN universal machine* to be studied in depth later. We will prove later that there are more than 10^{154} such Boolean functions of 9 variables that can be programmed by a single CNN universal machine. This immensely large number is greater than the volume of the universe (10^{84} cm³, calculated as a sphere with a diameter of 10 thousand million light years) !

5.2. Boolean and Compressed Local Rules

Every CNN with 3x3 neighborhood or its generalization, the *CNN universal machine*, to be presented later which maps a static *binary input* image into a static *binary output* image has a unique CNN truth table representation consisting of 512 rows, each one mapping a Boolean expression involving 9 Boolean variables, into a “0” or a “1” digit:

$$(d_1, d_2, \dots, d_9) \rightarrow \{0, 1\} \quad (5.4)$$

where $d_i \in \{0, 1\}$. We can now define rigorously our earlier heuristic notation of a local rule:

Definition 1: Complete Set of CNN Boolean Local Rules

Each row of the CNN truth table is called a Boolean *CNN local rule*. Every CNN with 3x3 neighbors is rigorously defined by a complete set of 512 Boolean Local Rules.

Definition 2: Compressed Boolean Local Rules

Any other rule which can be used to derive one or more Boolean local rules is called a *compressed local rule*, or simple local rule if the usage is clear.

The motivation for devising compressed local rules is simply to reduce the large number (512) of Boolean local rules to a smaller number. They are usually derived by heuristic methods and may not be adequate in view of the following reasons:

(a) While some local rule may correctly reproduce a large subset of the Boolean local rules, it may contradict some others. In this case, we say it is an *inconsistent* local rule. If the inconsistency occurs only for a few rare input patterns, it may still be useful for pedagogical purpose especially if the

local rule Compression Ratio $\gamma_{LR} \triangleq$ number of correctly reproduced Boolean local rules divided by 512.

is sufficiently large, $\gamma_{LR} \leq 1$. In this case the errors may be acceptable for pedagogical reasons especially if this local rule makes it possible to visualize or identify the main features of the input image that are to be extracted, modified, or transformed.

(b) The set of compressed local rules are *incomplete* in the sense that some Boolean local rules can *not* be deduced from them.

Definition 3: *Complete Set of Compressed Local Rules.*

A set of compressed local rules is said to be *complete* if and only if no member of this set is inconsistent and if all 512 Boolean local rules can be deduced from this set.

Definition 4: *Minimal Set of Compressed Local Rules.*

A complete set of compressed local rules is said to be *minimal* if no member of this set can be eliminated and still achieve completeness.

Computer-Aided Method for Proving Local Rules

Given a CNN template (A^0, B, z) and initial state $\mathbf{X}(0)$, there is presently no systematic algorithm to derive a *complete* set

$$\mathcal{L}_{LR} = \{ \mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_p \} \quad (5.5)$$

of local rules which are sufficient to map any binary input patterns into the prescribed output patterns obtained by solving the associated system of ODE's. In most cases, only a subset $\mathcal{L}_{LR}^- \subset \mathcal{L}_{LR}$ may be found. On rare occasions, a superset $\mathcal{L}_{LR}^+ \supset \mathcal{L}_{LR}$ may be found. On few occasions, some local rules may be *redundant* in the sense that for *some* input patterns, they predict the same output. It is also quite possible that two or more local rules may contradict each other's prediction and hence are said to be *inconsistent*. Finally, given a *complete* set of local rules, does there exist a proper subset which is also complete? If so, is it possible to find a complete set of local rules which are minimal in the sense that no other complete set exists which contains a fewer number of elementary local rules? We will now show that all of these questions, except the last one, can be easily resolved with the help of the CNN truth table, or equivalently, its associated minimal truth table. We will give a constructive solution to each question (except the last one) raised above in the form of an algorithm.

Algorithm 1: Checking whether a local rule candidate \mathcal{L}_i is consistent

1. Use the prescribed template (A^0, B, z) and initial state $\mathbf{X}(0)$ to derive the associated CNN truth table \mathcal{I} .
2. Apply the local rule δ_1 to *each* of the 512 input patterns. In general, δ_1 may *not* be *applicable* (NA) for some patterns (due to inadequate or overly simplistic assumptions). In this case, the output cell will be denoted by NA, or simply coded in gray scale. For those input patterns where δ_1 is applicable, there are 3 possibilities for the output cell: (i) output is black (coded by Boolean number 1) and agrees with the corresponding output in the truth table. In this case, the output will be printed “black”. (ii) output is white (coded by Boolean number 0) and agrees with the corresponding output in the truth table. In this case, the output cell will be printed “white”. (iii) the output is black (resp. white) but the corresponding cell in the truth table is white (resp., black). In this case, the output cell will be denoted by a cross \boxtimes , thereby indicating δ_1 is inconsistent and is not a valid local rule.
1. The local rule δ_1 is proved to be valid if and only if it is *not* inconsistent.

Algorithm 2: Checking whether a set $\delta = \{ \delta_1, \delta_2, \dots, \delta_k \}$ is complete.

1. Derive the CNN truth table \mathcal{I} , as in Algorithm 1.
2. Apply *Algorithm 1* to each $\delta_i \in \delta$. If any δ_i is inconsistent, stop. Otherwise, go to 3.
3. If *each* output cell is predicted to be either black or white by at least one local rule $\delta_i \in \delta$, then δ is complete. In this case, we have a rigorous proof of the validity *and* completeness of the set of local rules.

Algorithm 3: Given a complete set δ_{LR} of local rules, find a smallest proper subset which is also complete.

1. Delete δ_1 from δ_{LR} and apply Algorithm 2 to the remaining set. If it is complete, delete the first 2 elements δ_1 and δ_2 from δ_{LR} and repeat Algorithm 2. Continue the same “pruning” procedure until the remaining set is no longer complete. In this case, the immediately preceding remaining set of local rules constitutes the *smallest* complete set with respect to the order where the elements of δ_{LR} are deleted.
2. Repeat step 1 to all permutations of the ordering of the members of δ_{LR} .
3. Any complete set resulting from steps 1 and 2 having the smallest number of elements is a *minimal complete* set, relative to δ_{LR} .

Remarks:

1: The above choice of minimal complete set may not be unique. Since there may exist several complete sets all containing the same smallest number of elements.

2: The “minimality” derived from Algorithm 3 may not be global in the sense that there may exist an entirely different set δ_{LR} of complete rules in which Algorithm 3 would yield a minimal complete set having fewer elements than that determined from δ_{LR} . The difficulty in deriving a global minimal complete set is that there is no obvious algorithm to guarantee all distinct sets of complete local rules have been exhausted. A further difficulty lies in the criterion to be used for certifying which local rule is qualified as *elementary*. For otherwise, one could combine several local rules into a single but more complex local rule. Hence it is necessary to define “elementary” in the sense that no decomposition into two or more simpler local rules is possible. The algorithms are contained in the TEMPO program (Appendix C)

5.2. Optimizing the Truth Table

Recall that once a CNN template is specified a unique truth table can be easily generated by a simple computer program, say by solving a system of 9 ODE's a total of 512 times, one for each distinct Boolean pattern of nine input variables, or by some explicit formula that applies only to some specific subclass of CNN's, e.g., the *uncoupled* class. One can examine each of the 512 3×3 binary input patterns and determine whether the output (black or white) of this CNN is “correct” from the user's perspective. The next tables (Table 1-11) show Minimal Truth Table, the Truth Tables and the Window Truth Tables of the CORNER template. However, for example, among the 32 input patterns shown in Table 5 (corresponding to the Boolean local rules no. 96-127) and the 32 input patterns shown in Table 6 (corresponding to the Boolean local rules no. 160-191) for the CORNER CNN, we found the “black” output of this CNN for input patterns no. 114, 116, 176, 177, 178, 180, and 184 to be “incorrect” in the sense that the center black pixel in each of these 7 input patterns do *not look* like “corners”, from the perspective of the human visual system. Similarly, we also disagree with this CNN's classification (white; i.e., *not corner*) of input patterns no. 115 and 121 because the black center pixel in this two patterns really look like a “corner”. Hence, we would consider these 9 classifications made by the CORNER CNN to be “incorrect”. It is important that this does *not* mean the CORNER truth table is incorrect, as *every truth table* is an exact and hence correct representation of the CNN having the prescribed template. Indeed, from the perspective of a *robot*, or some creatures having a different visual system, the above classifications may be completely acceptable.

From the human perspective, however, it would be desirable to reclassify the above 9 Boolean local rules to obtain an optimized CNN truth table⁴. Once this is done, our next task is to design a CNN template (which may not exist) having this optimized

⁴ This reclassification task is a *subjective* exercise since not everyone may agree on whether a particular pixel in fuzzy cases is a corner, or not a corner.

truth table. If no such template exists, we will show later that a CNN universal machine can always be used to realize this optimized truth table, or any other truth table.

As an example, all misclassified input patterns by the CORNER CNN are designated in the minimal truth table shown in Table 12 by a *light-gray* pixel, if this pixel should be reclassified as white, and by a *dark-gray* pixel, if this pixel should be reclassified as black. The resulting *optimized* CORNER CNN is shown in Table 13. We leave this as a challenge to the reader invent a CNN template having this optimized CORNER truth table. Tables 14 and 15 show the binary and decimal code for the CORNER and optimized CORNER templates, respectively.

Minimal Truth Table of CORNER Template

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1																																
2																																
3																																
4																																
5																																
6																																
7																																
8																																
9																																
10																																
11																																
12																																
13																																
14																																
15																																
16																																

Table 1

[illegible]

Tables of Input-Output patterns for CORNER template (1, 2)

































































































































input 0:  output 0: 	input 1:  output 1: 	input 2:  output 2: 	input 3:  output 3: 	input 4:  output 4: 	input 5:  output 5: 	input 6:  output 6: 	input 7:  output 7: 
input 8:  output 8: 	input 9:  output 9: 	input 10:  output 10: 	input 11:  output 11: 	input 12:  output 12: 	input 13:  output 13: 	input 14:  output 14: 	input 15:  output 15: 
input 16:  output 16: 	input 17:  output 17: 	input 18:  output 18: 	input 19:  output 19: 	input 20:  output 20: 	input 21:  output 21: 	input 22:  output 22: 	input 23:  output 23: 
input 24:  output 24: 	input 25:  output 25: 	input 26:  output 26: 	input 27:  output 27: 	input 28:  output 28: 	input 29:  output 29: 	input 30:  output 30: 	input 31:  output 31: 
input 32:  output 32: 	input 33:  output 33: 	input 34:  output 34: 	input 35:  output 35: 	input 36:  output 36: 	input 37:  output 37: 	input 38:  output 38: 	input 39:  output 39: 
input 40:  output 40: 	input 41:  output 41: 	input 42:  output 42: 	input 43:  output 43: 	input 44:  output 44: 	input 45:  output 45: 	input 46:  output 46: 	input 47:  output 47: 
input 48:  output 48: 	input 49:  output 49: 	input 50:  output 50: 	input 51:  output 51: 	input 52:  output 52: 	input 53:  output 53: 	input 54:  output 54: 	input 55:  output 55: 
input 56:  output 56: 	input 57:  output 57: 	input 58:  output 58: 	input 59:  output 59: 	input 60:  output 60: 	input 61:  output 61: 	input 62:  output 62: 	input 63:  output 63: 

Table 4

Tables of Input-Output patterns for CORNER template (3, 4)

































































































































































































input 64:  output 64: 	input 65:  output 65: 	input 66:  output 66: 	input 67:  output 67: 	input 68:  output 68: 	input 69:  output 69: 	input 70:  output 70: 	input 71:  output 71: 
input 72:  output 72: 	input 73:  output 73: 	input 74:  output 74: 	input 75:  output 75: 	input 76:  output 76: 	input 77:  output 77: 	input 78:  output 78: 	input 79:  output 79: 
input 80:  output 80: 	input 81:  output 81: 	input 82:  output 82: 	input 83:  output 83: 	input 84:  output 84: 	input 85:  output 85: 	input 86:  output 86: 	input 87:  output 87: 
input 88:  output 88: 	input 89:  output 89: 	input 90:  output 90: 	input 91:  output 91: 	input 92:  output 92: 	input 93:  output 93: 	input 94:  output 94: 	input 95:  output 95: 
input 96:  output 96: 	input 97:  output 97: 	input 98:  output 98: 	input 99:  output 99: 	input 100:  output 100: 	input 101:  output 101: 	input 102:  output 102: 	input 103:  output 103: 
input 104:  output 104: 	input 105:  output 105: 	input 106:  output 106: 	input 107:  output 107: 	input 108:  output 108: 	input 109:  output 109: 	input 110:  output 110: 	input 111:  output 111: 
input 112:  output 112: 	input 113:  output 113: 	input 114:  output 114: 	input 115:  output 115: 	input 116:  output 116: 	input 117:  output 117: 	input 118:  output 118: 	input 119:  output 119: 
input 120:  output 120: 	input 121:  output 121: 	input 122:  output 122: 	input 123:  output 123: 	input 124:  output 124: 	input 125:  output 125: 	input 126:  output 126: 	input 127:  output 127: 

Table 5

Tables of Input-Output patterns for CORNER template (5, 6)

input 128:  output 128: 	input 129:  output 129: 	input 130:  output 130: 	input 131:  output 131: 	input 132:  output 132: 	input 133:  output 133: 	input 134:  output 134: 	input 135:  output 135: 
input 136:  output 136: 	input 137:  output 137: 	input 138:  output 138: 	input 139:  output 139: 	input 140:  output 140: 	input 141:  output 141: 	input 142:  output 142: 	input 143:  output 143: 
input 144:  output 144: 	input 145:  output 145: 	input 146:  output 146: 	input 147:  output 147: 	input 148:  output 148: 	input 149:  output 149: 	input 150:  output 150: 	input 151:  output 151: 
input 152:  output 152: 	input 153:  output 153: 	input 154:  output 154: 	input 155:  output 155: 	input 156:  output 156: 	input 157:  output 157: 	input 158:  output 158: 	input 159:  output 159: 

































































































































input 160:  output 160: 	input 161:  output 161: 	input 162:  output 162: 	input 163:  output 163: 	input 164:  output 164: 	input 165:  output 165: 	input 166:  output 166: 	input 167:  output 167: 
input 168:  output 168: 	input 169:  output 169: 	input 170:  output 170: 	input 171:  output 171: 	input 172:  output 172: 	input 173:  output 173: 	input 174:  output 174: 	input 175:  output 175: 
input 176:  output 176: 	input 177:  output 177: 	input 178:  output 178: 	input 179:  output 179: 	input 180:  output 180: 	input 181:  output 181: 	input 182:  output 182: 	input 183:  output 183: 
input 184:  output 184: 	input 185:  output 185: 	input 186:  output 186: 	input 187:  output 187: 	input 188:  output 188: 	input 189:  output 189: 	input 190:  output 190: 	input 191:  output 191: 

Table 6

Tables of Input-Output patterns for CORNER template (7, 8)

input 192:  output 192: 	input 193:  output 193: 	input 194:  output 194: 	input 195:  output 195: 	input 196:  output 196: 	input 197:  output 197: 	input 198:  output 198: 	input 199:  output 199: 
input 200:  output 200: 	input 201:  output 201: 	input 202:  output 202: 	input 203:  output 203: 	input 204:  output 204: 	input 205:  output 205: 	input 206:  output 206: 	input 207:  output 207: 
input 208:  output 208: 	input 209:  output 209: 	input 210:  output 210: 	input 211:  output 211: 	input 212:  output 212: 	input 213:  output 213: 	input 214:  output 214: 	input 215:  output 215: 
input 216:  output 216: 	input 217:  output 217: 	input 218:  output 218: 	input 219:  output 219: 	input 220:  output 220: 	input 221:  output 221: 	input 222:  output 222: 	input 223:  output 223: 

































































































































input 224:  output 224: 	input 225:  output 225: 	input 226:  output 226: 	input 227:  output 227: 	input 228:  output 228: 	input 229:  output 229: 	input 230:  output 230: 	input 231:  output 231: 
input 232:  output 232: 	input 233:  output 233: 	input 234:  output 234: 	input 235:  output 235: 	input 236:  output 236: 	input 237:  output 237: 	input 238:  output 238: 	input 239:  output 239: 
input 240:  output 240: 	input 241:  output 241: 	input 242:  output 242: 	input 243:  output 243: 	input 244:  output 244: 	input 245:  output 245: 	input 246:  output 246: 	input 247:  output 247: 
input 248:  output 248: 	input 249:  output 249: 	input 250:  output 250: 	input 251:  output 251: 	input 252:  output 252: 	input 253:  output 253: 	input 254:  output 254: 	input 255:  output 255: 

Table 7

Tables of Input-Output patterns for CORNER template (9, 10)

input 256:  output 256: 	input 257:  output 257: 	input 258:  output 258: 	input 259:  output 259: 	input 260:  output 260: 	input 261:  output 261: 	input 262:  output 262: 	input 263:  output 263: 
input 264:  output 264: 	input 265:  output 265: 	input 266:  output 266: 	input 267:  output 267: 	input 268:  output 268: 	input 269:  output 269: 	input 270:  output 270: 	input 271:  output 271: 
input 272:  output 272: 	input 273:  output 273: 	input 274:  output 274: 	input 275:  output 275: 	input 276:  output 276: 	input 277:  output 277: 	input 278:  output 278: 	input 279:  output 279: 
input 280:  output 280: 	input 281:  output 281: 	input 282:  output 282: 	input 283:  output 283: 	input 284:  output 284: 	input 285:  output 285: 	input 286:  output 286: 	input 287:  output 287: 

































































































































input 288:  output 288: 	input 289:  output 289: 	input 290:  output 290: 	input 291:  output 291: 	input 292:  output 292: 	input 293:  output 293: 	input 294:  output 294: 	input 295:  output 295: 
input 296:  output 296: 	input 297:  output 297: 	input 298:  output 298: 	input 299:  output 299: 	input 300:  output 300: 	input 301:  output 301: 	input 302:  output 302: 	input 303:  output 303: 
input 304:  output 304: 	input 305:  output 305: 	input 306:  output 306: 	input 307:  output 307: 	input 308:  output 308: 	input 309:  output 309: 	input 310:  output 310: 	input 311:  output 311: 
input 312:  output 312: 	input 313:  output 313: 	input 314:  output 314: 	input 315:  output 315: 	input 316:  output 316: 	input 317:  output 317: 	input 318:  output 318: 	input 319:  output 319: 

Table 8

Tables of Input-Output patterns for CORNER template (11, 12)

input 320:  output 320: 	input 321:  output 321: 	input 322:  output 322: 	input 323:  output 323: 	input 324:  output 324: 	input 325:  output 325: 	input 326:  output 326: 	input 327:  output 327: 
input 328:  output 328: 	input 329:  output 329: 	input 330:  output 330: 	input 331:  output 331: 	input 332:  output 332: 	input 333:  output 333: 	input 334:  output 334: 	input 335:  output 335: 
input 336:  output 336: 	input 337:  output 337: 	input 338:  output 338: 	input 339:  output 339: 	input 340:  output 340: 	input 341:  output 341: 	input 342:  output 342: 	input 343:  output 343: 
input 344:  output 344: 	input 345:  output 345: 	input 346:  output 346: 	input 347:  output 347: 	input 348:  output 348: 	input 349:  output 349: 	input 350:  output 350: 	input 351:  output 351: 

































































input 352:  output 352: 	input 353:  output 353: 	input 354:  output 354: 	input 355:  output 355: 	input 356:  output 356: 	input 357:  output 357: 	input 358:  output 358: 	input 359:  output 359: 
input 360:  output 360: 	input 361:  output 361: 	input 362:  output 362: 	input 363:  output 363: 	input 364:  output 364: 	input 365:  output 365: 	input 366:  output 366: 	input 367:  output 367: 
input 368:  output 368: 	input 369:  output 369: 	input 370:  output 370: 	input 371:  output 371: 	input 372:  output 372: 	input 373:  output 373: 	input 374:  output 374: 	input 375:  output 375: 
input 376:  output 376: 	input 377:  output 377: 	input 378:  output 378: 	input 379:  output 379: 	input 380:  output 380: 	input 381:  output 381: 	input 382:  output 382: 	input 383:  output 383: 

Table 9

Tables of Input-Output patterns for CORNER template (13, 14)

































































































































input 384:  output 384: 	input 385:  output 385: 	input 386:  output 386: 	input 387:  output 387: 	input 388:  output 388: 	input 389:  output 389: 	input 390:  output 390: 	input 391:  output 391: 
input 392:  output 392: 	input 393:  output 393: 	input 394:  output 394: 	input 395:  output 395: 	input 396:  output 396: 	input 397:  output 397: 	input 398:  output 398: 	input 399:  output 399: 
input 400:  output 400: 	input 401:  output 401: 	input 402:  output 402: 	input 403:  output 403: 	input 404:  output 404: 	input 405:  output 405: 	input 406:  output 406: 	input 407:  output 407: 
input 408:  output 408: 	input 409:  output 409: 	input 410:  output 410: 	input 411:  output 411: 	input 412:  output 412: 	input 413:  output 413: 	input 414:  output 414: 	input 415:  output 415: 
input 416:  output 416: 	input 417:  output 417: 	input 418:  output 418: 	input 419:  output 419: 	input 420:  output 420: 	input 421:  output 421: 	input 422:  output 422: 	input 423:  output 423: 
input 424:  output 424: 	input 425:  output 425: 	input 426:  output 426: 	input 427:  output 427: 	input 428:  output 428: 	input 429:  output 429: 	input 430:  output 430: 	input 431:  output 431: 
input 432:  output 432: 	input 433:  output 433: 	input 434:  output 434: 	input 435:  output 435: 	input 436:  output 436: 	input 437:  output 437: 	input 438:  output 438: 	input 439:  output 439: 
input 440:  output 440: 	input 441:  output 441: 	input 442:  output 442: 	input 443:  output 443: 	input 444:  output 444: 	input 445:  output 445: 	input 446:  output 446: 	input 447:  output 447: 

Table 10

Tables of Input-Output patterns for CORNER template (15, 16)

































































































































input 448:  output 448: 	input 449:  output 449: 	input 450:  output 450: 	input 451:  output 451: 	input 452:  output 452: 	input 453:  output 453: 	input 454:  output 454: 	input 455:  output 455: 
input 456:  output 456: 	input 457:  output 457: 	input 458:  output 458: 	input 459:  output 459: 	input 460:  output 460: 	input 461:  output 461: 	input 462:  output 462: 	input 463:  output 463: 
input 464:  output 464: 	input 465:  output 465: 	input 466:  output 466: 	input 467:  output 467: 	input 468:  output 468: 	input 469:  output 469: 	input 470:  output 470: 	input 471:  output 471: 
input 472:  output 472: 	input 473:  output 473: 	input 474:  output 474: 	input 475:  output 475: 	input 476:  output 476: 	input 477:  output 477: 	input 478:  output 478: 	input 479:  output 479: 
input 480:  output 480: 	input 481:  output 481: 	input 482:  output 482: 	input 483:  output 483: 	input 484:  output 484: 	input 485:  output 485: 	input 486:  output 486: 	input 487:  output 487: 
input 488:  output 488: 	input 489:  output 489: 	input 490:  output 490: 	input 491:  output 491: 	input 492:  output 492: 	input 493:  output 493: 	input 494:  output 494: 	input 495:  output 495: 
input 496:  output 496: 	input 497:  output 497: 	input 498:  output 498: 	input 499:  output 499: 	input 500:  output 500: 	input 501:  output 501: 	input 502:  output 502: 	input 503:  output 503: 
input 504:  output 504: 	input 505:  output 505: 	input 506:  output 506: 	input 507:  output 507: 	input 508:  output 508: 	input 509:  output 509: 	input 510:  output 510: 	input 511:  output 511: 

Table 11

Corrected Minimal Truth Table of CORNER template

- ◼ squares correspond to corner misclassification (they should be white)
- ◻ squares correspond to non-corner misclassification (they should be black)

[illegible]

Table 12

Optimized Minimal Truth Table of CORNER Template

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
1																																	
2																																	
3																																	
4																																	
5																																	
6																																	
7																																	
8																																	
9																																	
10																																	
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

Table 13

Binary Code for ***CORNER*** Template
(512 bits)

00000000,00000000,00000000,00000000,00000000,00000001,00000000,00000000
,00000000,00000001,00000000,00000000,00000001,00010111,00000000,00000000
,00000000,00000001,00000000,00000000,00000001,00010111,00000000,00000000
,00000001,00010111,00000000,00000000,00010111,01111111,00000000,00000000
,00000000,00000001,00000000,00000000,00000001,00010111,00000000,00000000
,00000001,00010111,00000000,00000000,00010111,01111111,00000000,00000000
,00000001,00010111,00000000,00000000,00010111,01111111,00000000,00000000
,00010111,01111111,00000000,00000000,01111111,11111111,00000000,00000000

Decimal Code for ***CORNER*** Template
(140 digits)

47,634,102,646,527,572,675,971
,460,498,910,645,354,219,674,273,748,634,236,474,670
,546,006,561,432,941,907,354,541,093,642,727,873,594
,350,604,011,030,198,552,062,948,695,326,343,495,680

Table 14

Binary Code for *Optimized CORNER* Template
(512 bits)

,00000000,00000001,00000000,00000000,00000000,00000001,00000000,00000000
,00000000,00000001,00000000,00000000,00000000,00000001,00000000,00000000
,00000000,00000001,00000000,00000000,00000000,00000101,00000000,00000000
,00000000,01010101,00000000,00000000,00000001,01010101,00000000,00000000
,00000000,00000001,00000000,00000000,00000011,00000011,00000000,00000000
,00000000,00010001,00000000,00000000,00000011,00110011,00000000,00000000
,00000000,00000111,00000000,00000000,00000111,00001111,00000000,00000000
,00000000,01011111,00000000,00000000,11111111,11111111,00000000,00000000

Decimal Code for *Optimized CORNER* Template
(150 digits)

204,586,913,041,142,969,522,351,928,009,830
,941,404,290,185,269,210,065,083,499,186,859,428,943
,804,165,897,630,843,608,945,882,697,576,708,597,045
,469,082,137,675,717,688,639,024,082,912,326,647,808

Table 15