

The Art of CNN Template Design

Ákos Zarándy

Computer and Automation Institute of the Hungarian Academy of Science
(MTA-SzTAKI), 11. Lágymányosi út, Budapest, H-1111, Hungary,
e-mail: zarandy@sztaki.hu,

ABSTRACT: *A practical survey of the design rules of uncoupled and coupled linear CNN templates with binary inputs and outputs is given. The usage and the properties of the different classes of CNN templates are analyzed. CNN chip specific robustness considerations are also given.*

1. Introduction

A Cellular Neural Network (CNN) [1,2] is a locally interconnected analog processor array arranged to regular 2D grid. Its two dimensional inputs and output makes it extremely suitable for image processing. Due to its regular (in most cases rectangular) arrangement and its local interactions, programmable CNN (the so called CNN Universal Machine [3]) can be efficiently implemented on silicon. With the today available deep submicron technology (0.33 μm - 0.25 μm) 64 \times 64 or up to 100 \times 100 large analog processor array can be implemented on a single chip. Some smaller operational test chips are already available [5,6,7]. The spatial-temporal transient of an analog VLSI CNN array settles down in the 100ns range. This means, that an image processing primitive (like edge detection, blurring, sharpening, thresholding, etc.) can be calculated for a 64 \times 64 (or 100 \times 100) pixel sized image in less than a microsecond with a single chip. In the digital word this incredible huge computational power is equivalent with Tera (10^{12}) operations per second [4], which is the same as the total computational power of a supercomputer containing ~9000 pieces of Pentium 200MHz. But while programming a digital microprocessor is relatively easy, here one has to find the exact parameter set of a continuous time spatial-temporal non-linear array dynamic to force it to calculate an image processing primitive. Since CNN has space invariant local interconnection structure it has 19 free parameters only. This parameter set, called *template*, exclusively determines its array dynamic behavior. This paper surveys the design methods of the different templates.

There are three major template design methods: (i) intuitive way, (ii) template learning, and (iii) direct template derivation. The first requires intuitive thinking of the designer. In several simple cases it leads to quick results. But on the other hand, it does not guarantee to find the desired template. Moreover designers need to have lots of experiments in both the image processing and the array dynamics.

The second design method, the template learning, is an extensively studied, popular field of the CNN research. Almost all classical neural network training methods have

been adopted to the CNN structure. But there are three serious problems with these techniques and results. First of all, the learning is based on input and desired output pairs, and during the learning procedure better and better results are supposed to be generated with better and better templates. But in many cases (especially propagating type templates like CCD [8]) the template either works or does not work, and there are no subsequent better and better result series. This might change the strategy of a learning method to a brute-force method. The second problem is, that in some cases template for the given problem does not exist. The learning methods cannot even realize it, and they run forever. If the template exists, it might take a long time to find it, if it can at all. The third problem is, that several proposed learning methods were tested and proved to be efficient to those template groups, which can be directly derived from the exact function descriptions. In these cases it is unnecessary to use learning methods. However, there are some cases when the template learning is a very important design method. In those cases, where there is no explicit desired output exist, the direct template design is impossible. Texture segmentation is a good example of this case [9].

The third method, considered in this paper, is the direct template design. It can be applied when the desired function is exactly determined. The design methods depend on the particular template class. While the previous two methods result a single or a few working templates, here we get all of them. This provides the opportunity to choose the best, the most robust template among them. Moreover, this method needs only a small fraction of the computational power the template learning needs.

In this paper we give a survey of the design and the usage of the different template classes. For each class first, we describe the basic properties of the network. Then, we survey the typical problems, which can be solved using templates from the particular class. After this we discuss the template design method and show some design examples. Finally, a CNN chip implementation specific robustness analysis and other important notes are given. The structure of the paper is the following: in Section 2 the uncoupled templates, while in Section 3 the coupled templates are discussed.

Our main goal with this paper is to give a practical introduction of the template design. Instead of strictly proving all of our statements, we rather show some tricks and interesting useful solutions. Some parts of this paper (like the binary uncoupled template design was already discussed in different forms [10,11,12,15]). Here we give a summary, and show some practical examples. Other parts, like the coupled binary CNN template design are new results. All template design methods, presented in this paper, can be used in both the *Chua-Yang model* [1] and the *Full Signal Range (FSR) model* [13]. In case of the FSR model the center element of the derived template should be decreased with 1.

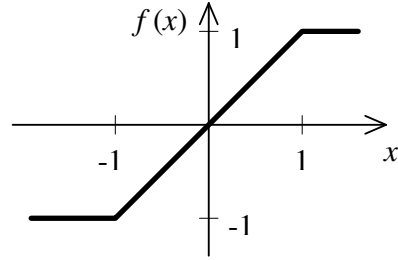
2. Uncoupled CNN templates

Uncoupled templates have zero off-center \mathbf{A} template elements only. The general form of the uncoupled templates is the following:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{00} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{-1-1} & b_{-10} & b_{-11} \\ b_{0-1} & b_{00} & b_{01} \\ b_{1-1} & b_{10} & b_{11} \end{bmatrix}, \quad z = i \quad (1)$$

Since their dynamic parts are uncoupled, they work as an array of independent first-order elements (cells). Hence, it is satisfactory to analyze the dynamic behavior of a single cell. A single cell receives 9 static inputs from its own input and from the input of the neighboring cells (u_{kl}). It has an initial condition $x(0)$, which can be considered as a tenth input. An uncoupled CNN cell maps these 10 inputs to a single output, with other words it implements a 10 input one output function. The state equation of a single cell is as follows:

$$\begin{aligned} \dot{x}(t) &= -x(t) + a_{00}y(t) + s \\ s &= \sum_{C(kl) \in N_r(i,j)} \mathbf{B}_{ij,kl} u_{kl} + z; \\ y &= f(x) \end{aligned} \quad (2)$$



Where: s is a constant during the transient evaluation, which depends on the template and the input of the CNN; f is the output characteristic, which is usually a sigmoid function. The flow-graph which describes the first order system of a single cell can be seen in Figure 1.

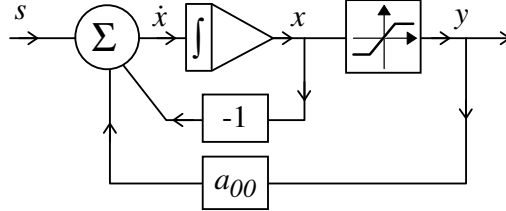


Figure 1. The flow-graph of the first order system of a single cell.

Before going on with the dynamic analysis of a single cell, let us describe the errors and deviations coming from the analog realization of the CNN. In case of the analog realization, the template parameters of the CNN are slightly varying. This can be considered as each cell has an individual template. The template values are close to the nominal which was downloaded to the whole array. The difference of the real template values and the nominal value changes from value to value and cell to cell, but they are constant in time. Their distribution and deviation depends on the particular chip design.

The dynamics of a cell depends on the a_{00} parameter. Instead of analyzing all the possible values of a_{00} we discuss the practically important cases. We call the following statements rules, and we will recall them later on this paper. These are as follows:

Rule 1. $a_{00}=0$. In this case the final output of the cell is equal to truncated contribution of the input, precisely: $y_{\infty}=f(s)$.

- Notes: 1. The final output is independent of the initial state $x(0)$.
2. The transient settles with an exponential decay.

Rule 2. $a_{00}=1$. In this case the CNN as an integrator, while x is in the linear region ($|x|<1$). If $s \neq 0$, than the final output depends on the sign of s , precisely: $y_{\infty} = \text{sign}(s)$. If $s=0$, than the final output depends on the initial state, precisely: $y_{\infty} = x(0)$.

- Notes: 1. If $s \neq 0$, than the final output is independent of the initial state $x(0)$, and it is binary.
2. The output saturates in less than $2s\tau$ time, where τ is the time constant of the cell.
3. In practice, we have to avoid the use of the later case($s=0$), because in case of an analog realization the precise equivalence is not a valid possibility hence, the final output will depend on the unpredictable parameter deviations.

Rule 3. $a_{00}>1$, usually 2 or even higher. The final output is always binary, due to the positive feedback loop. The final output depends on the initial state and the contribution of the input (s). There are three practically important cases:

- (i) $x(0)=0$ hence $y(0)=0$. The final output depends on the sign of s , precisely: $y_{\infty} = \text{sign}(s)$.
- (ii) $x(0)=+1$ hence $y(0)=+1$. The final output remains +1, if $a_{00}+s>1$. The final output changes to -1, if $a_{00}+s<1$. This will be called *Rule I*. in the next sections.
- (iii) $x(0)=-1$ hence $y(0)=-1$. The final output remains -1, if $-a_{00}+s<-1$. The final output changes to +1, if $-a_{00}+s>-1$. This will be called *Rule II*. in the next sections.
- (iv) $x(0)=x_0$, where $|x_0|<1$, hence $y(0)=x_0$. The final output depends on the sign of $\dot{x}(0)$, precisely: $y_{\infty} = \text{sign}(\dot{x}(0)) = \text{sign}(a_{00}x_0 - x_0 + s) = \text{sign}((a_{00}-1)x_0 + s)$. (This case is the generalization of the previous cases.)

Note: The $a_{00}+s=1$ situation and the $-a_{00}+s=-1$ situation should be avoided in practical cases, because in case of an analog realization the precise equivalence is not a valid possibility hence, the final output will depend on the unpredictable parameter deviations.

These rules can be trivially derived from the state equation (2) and the flow-graph (Figure 1.) of a single CNN cell. After analyzing the dynamic of a single cell, here we go through the four elementary uncoupled CNN template classes. In the following part of the

paper the inputs and the outputs of the CNN will be manifested in image forms. We follow the original convention of [1], where +1 stands for black, and -1 stands for white, and the intermediate values are represented by different gray shades.

2.1 Binary input - binary output uncoupled CNN templates

The uncoupled binary CNN templates form a very important class, because it covers the *binary image algebra or binary mathematical morphology* which is a well known and a frequently used tool-kit of the image processing. We grouped the most important templates into 3 classes. Here we show these groups and illustrate them with examples, list the template names from the Template Library [8] belonging to the each group, and introduce the design method.

Group I: Simple patter extraction

This group of templates extracts determined or underdetermined patterns. When describing the problem a binary 3×3 pattern and an integer number is given. The binary pattern contains black pixels, white pixels and “don’t care” pixels (See our example!). The integer number works as a limit. It says that at least how many positions of the pattern should match to extract (change black) a pixel. The black-and-white input image is placed to the input of the network. The output is black in those pixel positions, where there were at least as many matches as the integer number (limit number) indicates.

Design example 1:

Given the 3×3 binary pattern shown in Figure 2a. Suppose that the threshold value is 5. Figure 2 shows an example.

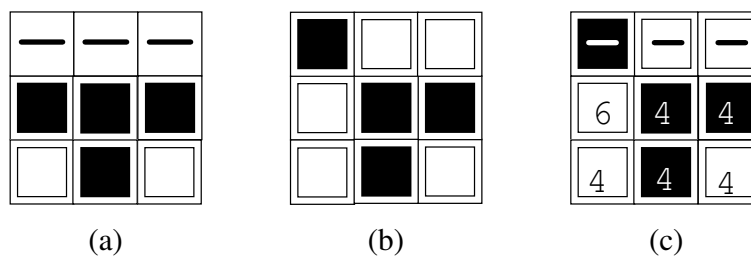


Figure 2. Example for the matching. (a) shows the binary pattern. Squares with ‘-’ means “don’t care”. (b) shows a test pattern. (c) shows the matching and the non-matching pixel locations. The matching positions are denoted with 4 and the non-matching one with 6. Since, there are 5 matching positions, the output of the cell will be black (+1).

Templates from the Template Library [8], which belong to this group:

EROSION, DILATION, DELVERT1, DIAG1LIU, FIGDEL, LSE, PEELHOR, RIGHTCON.

Note: in some cases more than one of these templates can be combined to a single template. ex: LCP.

Design method of the binary input-binary output templates

The design method of the templates belonging this class is illustrated with the flowchart shown in Figure 3. The first step is the most important, because the key of the successful template design is the correct template form determination. As we saw in (1), generally there are 11 free parameters of the uncoupled CNN templates. When we determine the template form, we drastically reduce the number of the free parameters. Some of the parameters will be set to zero, and some groups of it will be handled together. With this method the number of the free parameters is usually reduced to 3 or 4. This means, that in usual cases the template space is reduced to a 3 or 4 dimensional space. See (3) for the template form of the design example of Group 1!

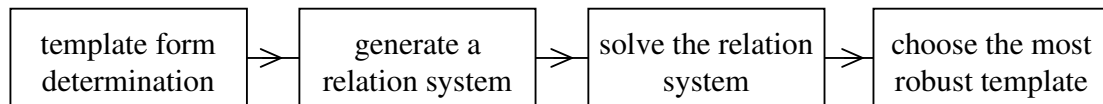


Figure 3. The flowchart of the design method of the binary input-binary output templates.

The second step of the template design is the generation of a relation system. It can be derived mechanically from the task and the *Rules*. Each relation guarantees the output to a certain input. Since the input output pairs are known, the generation of the relation system is simple. Each relation defines a hyper plane which cuts reduced template space into two halves. The relation is satisfied in one half and it is not in the other. Since all the relations should be satisfied, the intersection of the half spaces contains the correct templates. If it is an empty set the function cannot be solved with a single template (linearly not separable) in the determined template form. A graphical visualization example can be seen in Figure 4a. The solution method of the relation system is not in the scope of the paper. It can be solved by using any standard method, for example linear programming.

If we want to use a template with a CNN simulator we will see, that the convergence of some templates will be faster, others will be slower, but all templates in the specified template sub-space will work fine. But if we want to apply our template on a CNN chip we have to consider the parameter deviations coming from the analog realization. As we saw at the beginning of this section, the parameter deviation can be considered as each cell would have an individual template, which is close to the nominal template. Hence, the most robust template is in the middle of the specified template sub-space. It is shown graphically in Figure 4b. The nominal template is a single point, and the real templates belonging to the individual cells are in a circle around it.

Template form determination:

After the general idea of the design method was explained let us show it in practice with the example of *Group I*. First of all, the template form should be determined. The template form can be directly derived from the binary pattern (Figure 2a). a_{00} will be larger than 1 (say 2) which guarantee, that the final output will be binary (*Rule 3.*). The

initial condition will be zero, hence the final output will be $sign(s)$ (*Rule 3*). In the **B** template, the don't care positions are equal to zero. The black positions are playing the same roles, hence they can be characterized with the same free parameter, denote it by b . The role of the white positions are exactly the opposite of the black positions, hence they will not get a new free parameter. They will be $-b$. The template is sought in the following form:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ b & b & b \\ -b & b & -b \end{bmatrix}, \quad z = i \quad (3)$$

Relation system generation:

After determining the form of the template the generation of the relation system is easy. One has to go through all the possible combinations of the input pattern, and apply the particular Rules, in our case *Rule 3*. Numerically we can distinguish 7 different cases depending on the number of the matching pixels.

<i># of matching pixels</i>	<i>desired output</i>	<i>relation</i>	(4)
6	black (+1)	$6b+i>0$	
5	black (+1)	$4b+i>0$	
4	white (-1)	$2b+i<0$	
3	white (-1)	$i<0$	
2	white (-1)	$-2b+i<0$	
1	white (-1)	$-4b+i<0$	
0	white (-1)	$-6b+i<0$	

Solution of the relation system, and selection of the most robust template:

Here we are lucky, because there are only two free parameters of the system, hence we can solve the problem graphically. The graphical solution can be seen in Figure 4a. By solving the relation system, we got an infinite large subspace. From this subspace we have to pick a single point to be the nominal template. We have to consider the followings:

- It is a rule of thumb, that the more we scale up the template values, the faster the transient is.
- We suppose, that the real templates are in a fixed sized circle around the nominal template. To guarantee the robustness of the template this circle should be inside the specified subspace with its total volume. On the other hand, it can be seen, that the subspace opens (becomes wider) if the values are scaled up.
- The analog realization of the CNN always limits the maximal absolute value of the template values. Let say, that in our case the absolute value of a template element should not exceed 3, and the absolute value of the current should not exceed 6. It is the case in [7]. This limits the infinite subspace to a finite subspace. These boundaries are denoted with dashed lines in Figure 4b.

Hence, we have to choose the largest possible b and i values from the middle of the subspace. These values ($b=2.2, i=-6$) determines the selected template. We call the selected template as the nominal template. The real templates will be around it in the circle. The chosen best template is the following:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 2.2 & 2.2 & 2.2 \\ -2.2 & 2.2 & -2.2 \end{bmatrix}, \quad z = -6 \quad (5)$$

Notes:

1. The specialty of this template group is, that the \mathbf{B} template contains one free parameter only, hence it is constructed from zero, a certain real number and its opposite.
2. From the robustness point of view, it is more difficult to implement the template, if the threshold number is larger (6 instead of 5 in our case), because it makes the result template subspace narrower. If the result template subspace is narrower it might be difficult to place the circle with its total volume inside.

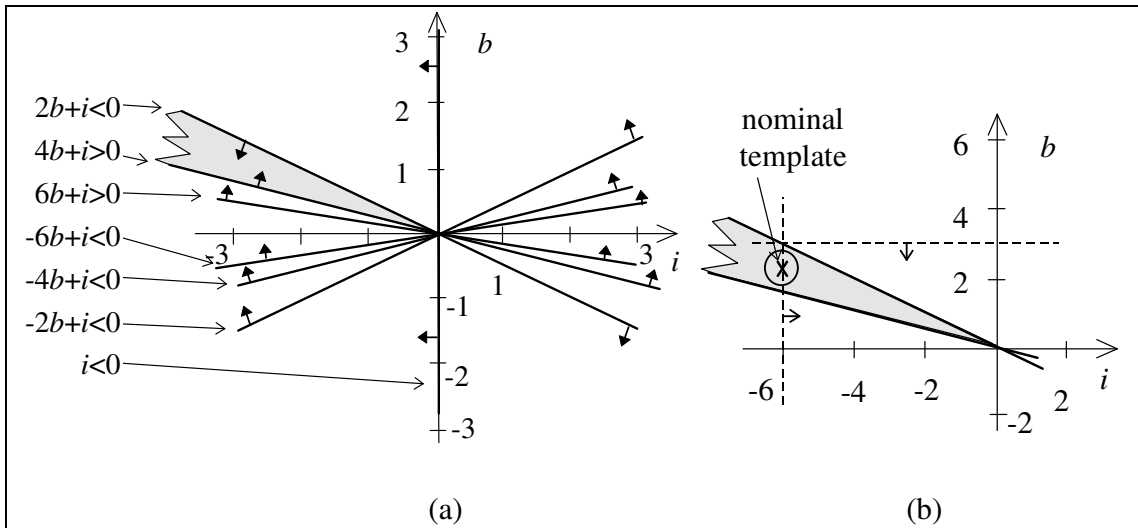


Figure 4. (a): Graphical solution of the relation system (4). The solution subspace is shaded. (b): Selection of the nominal template. The dashed lines shows the technical limitations of the CNN chip. The X shows the chosen best nominal template, and the circle around it contains the real templates.

Group II: Conditional pixel manipulation

While templates in the previous group redraw the whole image, here some pixels are changed and the rest of the picture is unchanged. In case of an asymmetric template ($z \neq 0$) either some black pixels change white or white pixels change black. In case of a

symmetrical template ($z=0$) if the condition is satisfied, black pixel goes to white, and if the opposite of the condition is satisfied, white pixel goes to black. But pixels in those locations, where neither the condition, nor its opposite is satisfied, the pixels remain unchanged. This is why it is called conditional pixel manipulation. Here given a binary pattern, a threshold (limit) number, and a rule whether to change white pixels to black or black pixels to white. When the number of the matching positions is calculated the off-center positions should be concerned only. In our example, an asymmetrical case will be shown, because it is more important from the practical point of view.

Design example 2:

Given the 3×3 binary pattern shown in Figure 5a. The task is the following: Change white those black pixels, whose neighborhood matches the given pattern in 5 positions, and let the other black and white pixels unchanged. Figure 5bc shows an example. (The resulting template will be the first template from the skeletonization template series [8].)

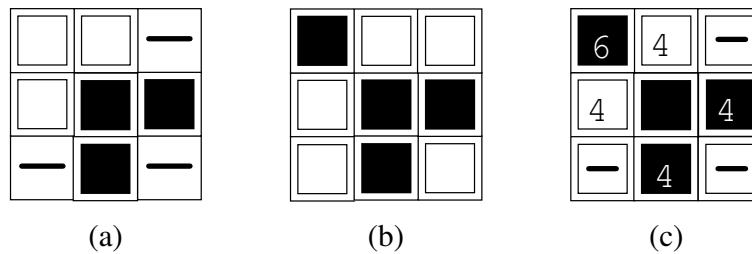


Figure 5. (a) is the given binary pattern. (b) is the test pattern. (c) shows the matching and the non-matching pixel locations. Since, there are 4 matching positions instead of 5, the output of the cell will not change. Note, that when the matching positions were calculated the center position was not concerned.

Template form determination:

The template form can be directly derived from the binary pattern (Figure 5a). a_{00} will be larger than 1 which guarantee, that the final output will be binary (*Rule 3*). The initial state will be the same as the input, hence the final output will be determined by *Rule 3*. In the **B** template, the don't care positions are equal to zero. The specialty of this group is, that the center position of the **B** template always plays a different role than the other positions. The reason is, that it stands for the satisfaction of the condition. (Recall, that the templates in this template group effected either the white or the black pixels in an image.) Hence, the center of the **B** template always gets an own free parameter, (say b). The black positions (not including the center position) play the same roles, hence they can be characterized by the same free parameter, name it c . The role of the white positions (not including the center position) is exactly the opposite of the black positions, hence they will not get a new free parameter. They will be $-c$. The template is sought in the following form:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -c & -c & 0 \\ -c & b & c \\ 0 & c & 0 \end{bmatrix}, \quad z = i \quad (6)$$

Relation system generation:

Since the initial state of the CNN is binary here and $a_{00} > 1$, we have to consider (ii) and (iii) of *Rule 3*. Here the number of the relations will be twice as many as in the previous case, because the individual cases depend not only on the number of the matching pixels, but also on the self input of the cell which can be either black or white. The relations are as follows:

<i>self input</i>	<i># of matching pixels</i>	<i>desired output</i>	<i>relation</i>	(7)
black (+1)	5	white (-1)	$a+b+5c+i < 1$	
black (+1)	4	black (+1)	$a+b+3c+i > 1$	
black (+1)	3	black (+1)	$a+b+c+i > 1$	
black (+1)	2	black (+1)	$a+b-c+i > 1$	
black (+1)	1	black (+1)	$a+b-3c+i > 1$	
black (+1)	0	black (+1)	$a+b-5c+i > 1$	
white (-1)	5	white (-1)	$-a-b+5c+i < -1$	
white (-1)	4	white (-1)	$-a-b+3c+i < -1$	
white (-1)	3	white (-1)	$-a-b+c+i < -1$	
white (-1)	2	white (-1)	$-a-b-c+i < -1$	
white (-1)	1	white (-1)	$-a-b-3c+i < -1$	
white (-1)	0	white (-1)	$-a-b-5c+i < -1$	

The solution of the relation system and the robust template selection is not in the scope of this paper. But there is one comment, what we have to emphasize here. Note, that a and b parameters has the same weights in each row of the relation set. Hence, the relation system remains satisfied, if we keep the sum of the a and b fix but play with the a and the b values, as far as $a > 1$. This sometimes help to squeeze the template element into the template range specified by the chip. For example, if we find a robust template, where the $a_{00}=2$, and $b_{00}=4$, and the rest of the template elements does not exceed the template limitation, instead of scaling down all the template elements (which is not good from the robustness point of view) we can simply change both a_{00} and b_{00} to 3.

Templates from the Template Library [8], which belong to this group:

Corner, Edge, Skeletonizing, Center, Figextr, Junction

Group III: Two input, one output functions

The specialty of this group is that the input and the initial state of the CNN is different, hence an additional input appears, and the total number of the pixels, which effect to the final output is 10 (instead of 9 like in the previous two groups). The image downloaded to the initial state of the network can be used as a mask. To the input image of the network the same spatial functions can be defined like we saw in the

previous two groups. The resulting image of this function and the initial state can be logically combined with the same template. For example this means, that a certain pattern is extracted from the input image, but the extracted black pixels appear only at those positions on the final output, where the initial state was black. For simplicity we show here a very simple example: the design of the AND template.

Design example 3:

Given two binary images. The final output is black in those positions, where both the input and the initial state was black, otherwise white.

Design considerations:

Here the input and the initial state holds different images. To guarantee, the final binary output a_{00} should be larger than 1 (*Rule 3.*). The **B** template form can be directly derived from the binary pattern, as it was shown in the previous two cases. The final output will be determined by *Rule 3.* The generation of the relation system is similar to the previous cases.

Templates from the Template Library [8], which belong to this group:

LOGAND, LOGDIF, LOGOR, LOGORN.

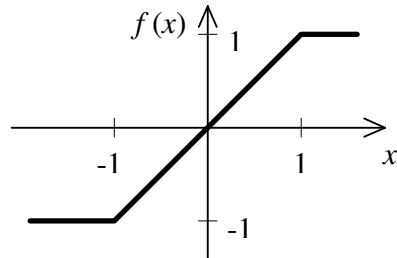
Comments:

Generally, all of these functions can be described as a 10 input one output logic function. Only a small fractions of these 1024 logic functions can be implemented with a single CNN template, the rest of them can be implemented with a few of templates. A theoretical analysis of this problem is described in [11].

3. Coupled CNN templates

A CNN with coupled template has an array dynamics. The change of the output of an individual cell effects its neighbors output and vice versa. The array dynamics is described by the following coupled first order differential equation system [1]:

$$\begin{aligned} \dot{x}(t) &= -x(t) + \sum_{C(kl) \in N_r(i,j)} \mathbf{A}_{ij,kl} y_{kl}(t) + s \\ s &= \sum_{C(kl) \in N_r(i,j)} \mathbf{B}_{ij,kl} u_{kl} + z; \\ y &= f(x) \end{aligned} \tag{8}$$



We separated the contribution of the **B** template and the bias term, because this term ‘s’ is constant during the spatio-temporal dynamics. In this section, we suppose, that $a_{00} > 1$.

3.1 Binary input - binary output coupled CNN templates

A Coupled Cellular Neural Network structure allows propagation phenomenon. The main questions in this section are, how this propagation happens, and how to control the propagation with proper template design. But before we start analyzing the propagation phenomena let us first define the notions of inactive and active cells, and the binary activation pattern.

A cell is considered to be inactive in a certain time instant, if its output is in a stable equilibrium point. An inactive cell must be in the saturation region, because due to the positive self feedback ($a_{00} > 1$), the cell cannot be in a stable equilibrium point in the linear region [1]. If one examines (8) carefully, he or she can find, that a cell is stable in the positive saturation region if the $\sum_{C(kl) \in N_r(i,j)} \mathbf{A}_{ij,kl} y_{kl}(t) + s$ value is larger than +1, or it is stable in the negative saturation region, if this value is smaller than -1.

A cell is considered to be active in a certain time instant, if its output is changing. An active cell is always in the linear region, and (in our cases) it always goes from one saturation region towards the other.

When analyzing the spatial temporal propagation behavior of a CNN array, we can separate the cells into the *active cell set* and the *inactive cell set* in every time instants. A cell must belong to one of these set. During a transient a cell might belong to the *active cell set*, then it changes its state and belongs to the *inactive cell set*, and this can be dynamically changing for each cells in time. When the transient decayed, all cells belong to the *inactive cell set*. Next, we analyze the run of a propagating type transient.

- At the beginning of a propagation type transient there are some active cells. If all the cells are active, we cannot really talk about a propagation type template. If there are no active cell at the very beginning, the transient can be considered to be settled.

- During the transient (not at the beginning) a cell can become active if and only if one of its direct active neighbor activates it. If a cell is active, its output is changing in time, hence the $\sum_{C(kl) \in N_r(i,j)} \mathbf{A}_{ij,kl} y_{kl}(t) + s$ term of its neighbors are changing. This change might activate some inactive neighbors. Hence, an inactive cell cannot become active, if it has no direct active neighbor. We deal only with those cases, when the activated cell goes from one saturation region to the other and becomes inactive there (at least for a while).

- At the end of the transient all cells are inactive.

Now, we are ready to analyze the propagation in pixel level via an example. Consider the binary image in Figure 6. Suppose, that we have a propagation type of template, which deletes the end of the single pixel line. In our example there is only one active cell at the beginning of the transient. It changes from black to white. When it changed, its neighbor becomes the end of the line, hence it becomes active, and starts changing to white, and so on... This is the way, how binary propagation type templates

work. Next, we show 3 property pairs of the propagation rules. Table 1 lists the propagation type templates from the Template Library [8] and shows their properties.

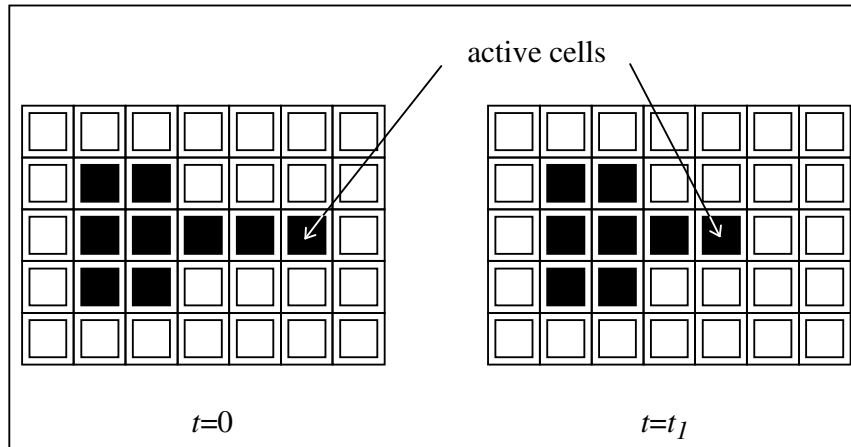
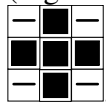


Figure 6. Example for the propagation. During the transient there is only one active cell at a time. The single pixel line is deleted pixel-by-pixel.

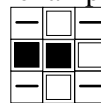
Statistical versa Pattern oriented propagation

A typical statistical ask is the following: in a 4 (or 8) neighborhood environment change those black (white) pixels white (black) which have at least 2 white (black) direct neighbors. In this case, the propagation front can move to any direction. Our example (Figure 6) can be described with the following rule: given the binary activation pattern:



. Change those black pixels white, which have at least 3 white neighbors (less than two matching positions). Applying this rule, all single pixel lines with arbitrary orientation will be deleted. Hence, this is not an orientation selective rule.

In case of pattern oriented propagation, the direction of the propagation is strictly determined by the binary activation pattern. Our example can be described with the



following rule: given the binary activation pattern: . Change a black pixels white if the whole binary pattern is matching in its location. Applying this rule, horizontal single pixel lines from right to left will be deleted only. Hence, this is an orientation selective rule. Note, that the pattern is asymmetric in the direction of propagation. This asymmetry allow the propagation and determines its direction.

Symmetrical versa Asymmetrical propagation

A propagation rule is symmetric, if the activation condition is symmetric to the color of a cell. For example, if the rule says that white cell changes black, if it has at least 3 black neighbor, than a black cell changes to white, if it has at least 3 white neighbor. A propagation type template is symmetric, if its current (z term) is zero.

Controlled versa Uncontrolled propagation

In case of a controlled template, the input contains a mask image. This mask image (which usually differs from the initial state) limits the propagation. For example, propagation can run over those areas which are black in the input image. If a propagation type template is controlled, its B template is not zero.

<i>template name</i>	<i>statistical</i>	<i>pattern oriented</i>	<i>symmetric</i>	<i>asymmetric</i>	<i>controlled</i>	<i>uncontrolled</i>
Average	4		4			4
CCD		4	4			4
CCDMASK		4		4	4	
Connectivity	4			4	4	
Erasmask		4		4	4	
Reconstruction	4		4		4	
Findarea	4			4	4	
Hole	4			4	4	
Shadowmask		4		4	4	
Shadow		4		4		4
Shadowsim		4		4		4
Hollow	4			4	4	
Patchmaker	4			4	4	
Smallkiller	4		4		4	

Table 1. The properties of the propagation type templates from the Template Library [8].

After analyzing the propagation types, we can show the design method. The flow-chart of the design steps of the propagation type templates can be seen in Figure 7.

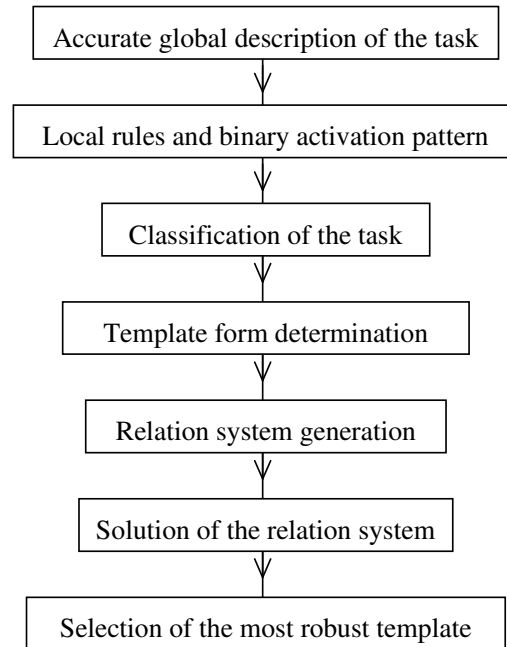


Figure 7. The flowchart of the design steps of the propagation type templates.

In the first step, we have to describe the global task verbally and with some input-output pairs. This description must be as precise, that based on this, one can generate the output from an arbitrary input image.

Next, we derive the local rules from the global description of the task. The pixel level rules of the propagation should be derive from the global task, like we did in our example in Figure 6. Then, we can generate the binary activation patterns.

In the classification step, we have to classify the task using the using the previously introduced clusters. Here it turns out, whether the propagation rule is statistical or pattern oriented, whether it is controlled or uncontrolled, whether it is symmetrical or asymmetrical. Investigating these questions helps us to execute the next step.

The center element of the **A** template is always a free parameter. The rest of the non don't care elements are the second free parameter. If some of them are white and the others are black, their sign are opposite in the template form. If the template is controlled, we have to introduce some new free parameters to the **B** template, according to the activation pattern. In most cases, it is one new free parameter in the center position of the **B** template, but sometimes (according to the input dependency of the activation patter) there can be one more additional free parameter in the neighborhood. If the task is asymmetric, the current brings a new free parameter.

The last three steps are the same as it was in the previous section. For illustrating the design method, here we show two design examples, a simple, and a very difficult:

Design example 4

Task: Generate the left to right horizontal shadow of the black object on a binary image.

1. *Global description*

This is a row-wise problem. If there is a black pixel in a row, all white pixels which are right to it should change black, and the rest of the pixels should be unchanged. An example can be seen in Figure 8.

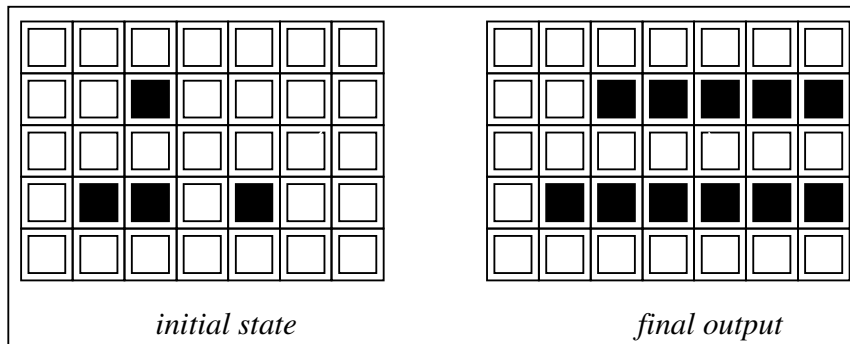
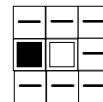
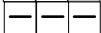


Figure 8. Example for the left to right shadow generation.

2. *Local rules and binary activation pattern*

In this task, we have to find the left most black pixel in each row, and change all white pixels black which are left to it. This can be done by starting a black propagation front moving right from each black pixels. Hence, the local rules are: (i) a white pixel, which direct left neighbor is black change black; (ii) the rest of the pixels should be unchanged.



The activation pattern belonging to this local rule is: . White pixel in this situation should change black.

3. *Classification*

The propagation is pattern oriented, because there is a single allowed propagation direction (left to right) only. It is uncontrolled, because the propagation is not limited. (It starts from the left most black pixel, and goes along to the boundary.) It is asymmetric, because the black objects get shadows, the white ones (here the background) do not.





4. *Template form determination*

The template form can be derived from the activation pattern and the classifications. The center element of the **A** template (a_{00}) is the first free parameter. There is only one off-center in the activation pattern, which is the second free parameter. The **B** template is zero, because the propagation is uncontrolled. The current (z) is the third free parameter, because the propagation is asymmetric. The template is sought in the following form:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ b & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad z = i \quad (9)$$

5. Relation system generation

Since two pixels effects the propagation, we have to examine for binary cases only. These cases are the following:

<i>local pixel arrangement</i>	<i>desired output</i>	<i>state</i>	<i>relation</i>	
	white	inactive	$-a-b+i < -1$	
	black	inactive	$a-b+i > 1$	(10)
	black	inactive	$a+b+i > 1$	
	black	active	$-a+b+i > -1$	

We do not deal with the last two steps here, because it is the same as it was in the uncoupled case.

Design example 5

Task: Given two binary images. The first contains some black objects against white background. The second is derived from the first one by changing some black pixels to white. In this way some objects become smaller in the second image than in the first. Those objects, which became smaller in the second image are considered to be marked. design a template, which deletes the marked objects and do not effect the rest of the image. If we delete a single pixel of a black object and apply this template, all the black pixels connected to the object will change white. Hence this template is called Connectivity in the Template Library.

1. Global description

This is a 2D problem. All connected black pixels of the marked objects should change white, and the rest of the pixels should be unchanged. An example can be seen in Figure 10.

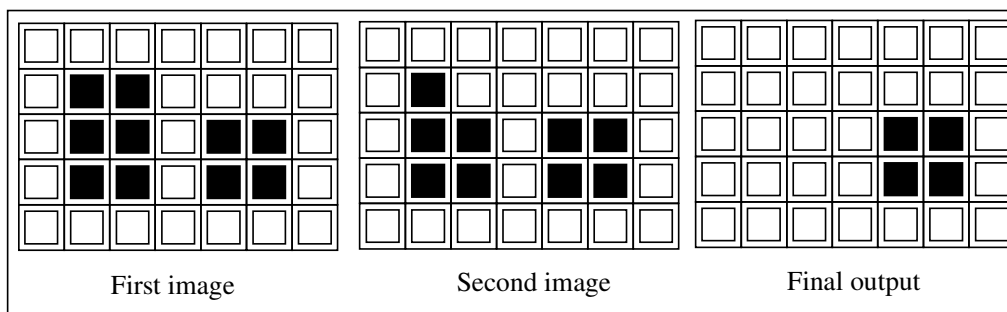


Figure 9. Example for the Connectivity template.

2. Local rules and binary activation pattern

In this task first, we have to find those pixels which are black in the first image and white in the second image. From these points we have to start propagation waves fronts to all directions. The front should propagate on the black pixels only and change them to white. Since the wave front moves on the second image, it will be the initial state and the first image will be the input. Hence, the local rules are the following: (i) change those black pixels to white which has at least one neighboring cell with white output and black input, and (ii) do not change the rest of the pixels. From this it follows that here the difference of the output and the input counts instead of simply the output value of the neighboring cells.

In this task the activation pattern is not a single 3×3 pattern, but two patterns, because the activation depends on both the output and the input. We introduced a new sign in the activation pattern. The delta sign (Δ) in a circle means that the particular neighbor activates the cell if and only if its output and its input is different. Note, that the definition of the task excludes those situations when the output is black and the input is white. A cell becomes active if it has at least one matching neighbor. For simplicity we used four neighborhood. The activation patterns can be seen in Figure 10.

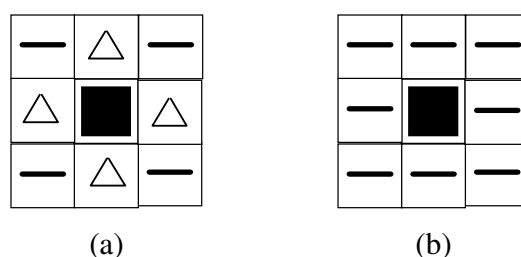


Figure 10. Binary activation patterns for the Connectivity template. (a) shows the output dependency of the activation and (b) shows the input dependency. The delta sign (Δ) means the different output and input.

3. Classification

The propagation is statistical, because the activation depends only the number of the matching pixels but not on their exact position. Hence, the propagation wave-form

can move to any of the four directions. It is controlled, because the propagation can go over the black areas only. It is asymmetric, because it deals with the black objects, the originally white pixels are unchanged.

4. Template form determination

As usual the template form can be derived from the activation pattern and the classifications. The center element of the **A** template (a_{00}) is the first free parameter. The delta operators in the neighborhood effects both the **A** template and the **B** template. A neighbor which has the same input and output (can be both black or white) does not effects the cell. But if it has black input and white output it activates the cell. Hence, the second free parameter appears in the neighborhood in both the **A** and the **B** template, but with opposite sign. The center element of the **B** template is the third free parameter. Since the propagation is asymmetric, the current (z) is the fourth free parameter. The template is sought in the following form:

$$\mathbf{A} = \begin{bmatrix} 0 & b & 0 \\ b & a & b \\ 0 & b & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & -b & 0 \\ -b & c & -b \\ 0 & -b & 0 \end{bmatrix}, \quad z = i \quad (11)$$

5. Relation system generation

Since there are only three valid binary input-output combinations here, and 5 matching possibilities, there are 15 different cases. All cases yield a relation. The relation set is the following:

output	input	# of matching pixels	state	desired output	relation	
black (+1)	black (+1)	0	active	black (+1)	$a+c+i>1$	
black (+1)	black (+1)	1	inactive	white (-1)	$a-2b+c+i<1$	
black (+1)	black (+1)	2	inactive	white (-1)	$a-4b+c+i<1$	
black (+1)	black (+1)	3	inactive	white (-1)	$a-6b+c+i<1$	
black (+1)	black (+1)	4	inactive	white (-1)	$a-8b+c+i<1$	
white (-1)	black (+1)	0	inactive	white (-1)	$-a+c+i<-1$	
white (-1)	black (+1)	1	inactive	white (-1)	$-a-2b+c+i<-1$	
white (-1)	black (+1)	2	inactive	white (-1)	$-a-4b+c+i<-1$	(12)
white (-1)	black (+1)	3	inactive	white (-1)	$-a-6b+c+i<-1$	
white (-1)	black (+1)	4	inactive	white (-1)	$-a-8b+c+i<-1$	
white (-1)	white (-1)	0	inactive	white (-1)	$-a-c+i<-1$	
white (-1)	white (-1)	1	inactive	white (-1)	$-a-2b+c+i<-1$	
white (-1)	white (-1)	2	inactive	white (-1)	$-a-4b-c+i<-1$	
white (-1)	white (-1)	3	inactive	white (-1)	$-a-6b-c+i<-1$	
white (-1)	white (-1)	4	inactive	white (-1)	$-a-8b-c+i<-1$	

We do not deal with the last two steps here, because it is the same as it was in the uncoupled case.

4. Conclusions

A classification and a systematic design method of the uncoupled and coupled binary input binary output CNN templates was introduced. Design examples for all template classes with robustness considerations were provided. The binary input binary output templates from the Template Library [8] were classified. The design method results the whole theoretically operational template set. From these infinity many templates, one can find the most robust one, or the fastest one or any optimum between them.

References

- [1] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory and Applications", *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, October 1988, pp. 1257-1290, 1988.
- [2] L.O. Chua and T. Roska, "The CNN Paradigm", *IEEE Transactions on Circuits and Systems - I*, vol. 40, no. 3, March 1993, pp. 147-156, 1993.
- [3] T. Roska and L.O. Chua, "The CNN Universal Machine: An Analogic Array Computer", *IEEE Transactions on Circuits and Systems - II*, vol. 40, March 1993, pp. 163-173, 1993.
- [4] L.O. Chua, T. Roska, T. Kozek, Á. Zarándy "CNN Universal Chips Crank up the Computing Power", *IEEE Circuits and Devices*, July 1996, pp. 18-28, 1996.
- [5] H.Harrer, J.A.Nosseck, T.Roska, L.O.Chua, "A Current-mode DTCNN Universal Chip", Proc. of IEEE Int. Symposium on Circuits and Systems, pp135-138, 1994.
- [6] J.M.Cruz, L.O.Chua, and T.Roska, "A Fast, Complex and Efficient Test Implementation of the CNN Universal Machine", Proc. of the third IEEE Int. Workshop on Cellular Neural Networks and their Application (CNNA-94), pp. 61-66, Rome Dec. 1994.
- [7] S.Espejo, R.Dominguez-Castro, A.Rodriguez-Vázquez and R.Carmona, "CNNUC2 User's guide", Centro Nacional de Microelectrónica, Seville, 1995.
- [8] "CNN Software Library, (Templates and Algorithms)", edited by T.Roska, L. Kék, L. Nemes, and Á. Zarándy, Comp. and Auto. Ins. of the Hung. Acad. of Sci. DNS-1-1997, Budapest, 1997.
- [9] T. Szirányi, M. Csapodi: "Texture Classification and Segmentation by Cellular Neural Network using Genetic Learning", (CVGIP) Computer Vision and Image Understanding, accepted, 1997.
- [10] Á. Zarándy, A. Stoffels, T. Roska, F. Werblin, and L.O. Chua "Implementations of Binary and Grayscale Mathematical Morphology on the CNN Universal Machine" paper accepted to IEEE Trans. Circuits and Systems, 1997.
- [11] L. Nemes, L.O. Chua, "Linear Decomposition of Linearly Non-separable Boolean Functions and their Implementation on CNN Universal Machine", paper submitted to International Journal of Circuit Theory and Applications, 1997.
- [12] L.O. Chua and T. Roska, "Cellular Neural Networks: Foundations and Primer", Lecture Notes for the course EE129 at U.C. Berkeley, 1997.
- [13] S.Espejo, A.Rodriguez-Vázquez, R.Dominguez-Castro, and R.Carmona "Convergence and Stability of FSR CNN Model" Proc. of the IEEE Conference on Cellular Neural Networks and their Applications (CNNA-96), pp. 411-416. Seville, 1996.
- [14] L. Kék, Á. Zarándy, "Implementation of Large Neighborhood Nonlinear templates on the CNN Universal Machine" paper submitted to International Journal of Circuit Theory and Applications, 1997
- [15] K.R.Crouse, E.L.Fung, L.O.Chua, "Efficient Implementation of Neighborhood Logic for Cellular Neural Network Universal machine", *IEEE Trans. Circuits and Systems I*. vol. 44. No.4. pp.255-361, April, 1997