Recognition of Car License Plates using a Neocognitron type of Artificial Neural Network

Bas Cornet, 2001 may.

Introduction

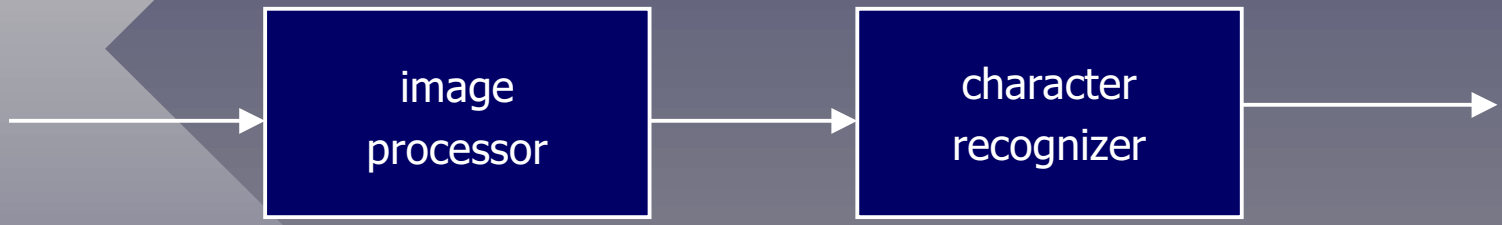
universal character reader

XNZD18

CRXU 121407

Introduction

The basic model



- Image processor
  - finding and selecting all possible character images at any location, from any size from any color
- Character recognizer
  - classifying all character images, with any style/font

Introduction

Scope limitation

image processor → character recognizer → JSPT84

- Image processor
  - traditional programming using a license plate definition
- Character recognizer
  - neural computing using a neocognitron

Presentation

Contents

1. Image processor	(5 sheets)
2. Neocognitron theory	(8 sheets)
3. Application in CLPR	(8 sheets)
4. CLPR system	(5 sheets)
5. Conclusion	(1 sheet)
Demonstration	(5 minutes)

Image processing

- What: Finding the possible characters
- How: Image transformations
- Not: Using explicit knowledge

Finding the plate characters

Original image

Contrast stretchted

Filtered

Finding the plate characters

Original image

Contrast stretchted

Filtered

Biniarized

Finding the plate characters

Original image

Contrast stretchted

Filtered

Biniarized

Area filled

Finding the plate characters

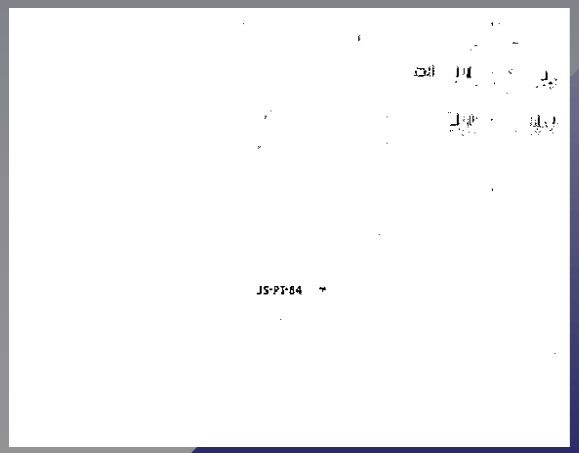
Original image

Contrast stretchted

Filtered

Biniarized

Area filled

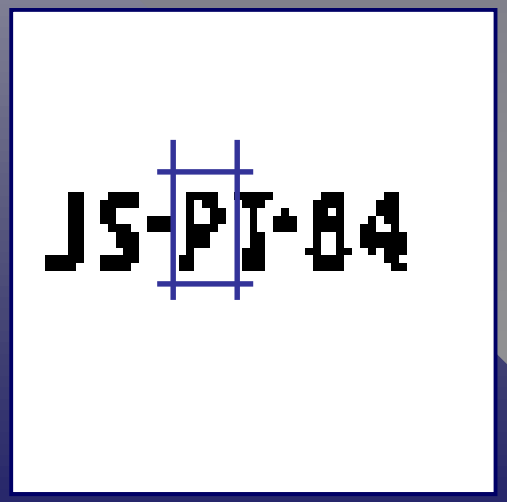


Pre-processed

Catching the segments

Original picture

Processed picture



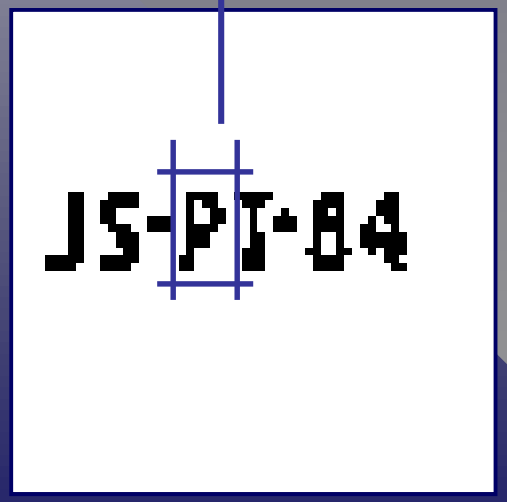
Segment Location

Catching the segments

Original picture

Processed picture

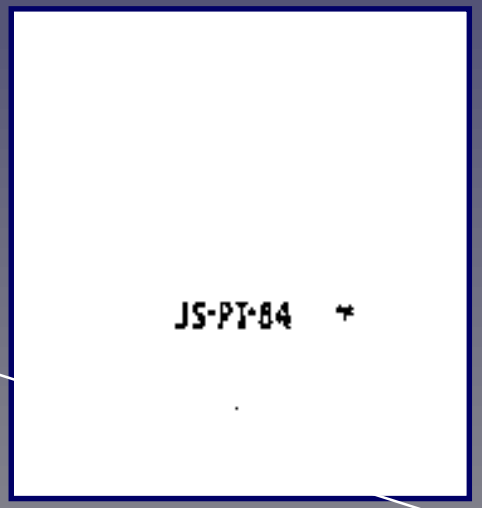
Segment Location



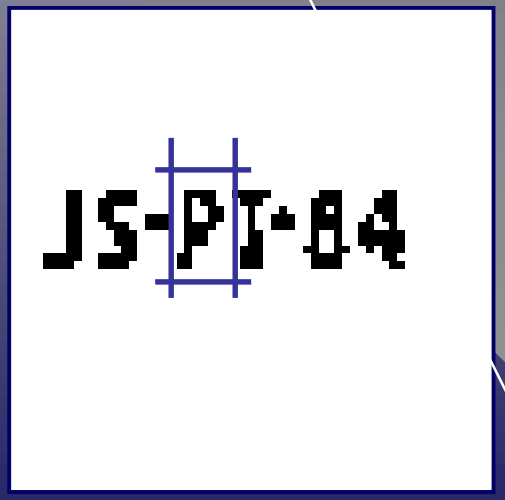
Catching the segments

Original picture

Processed picture



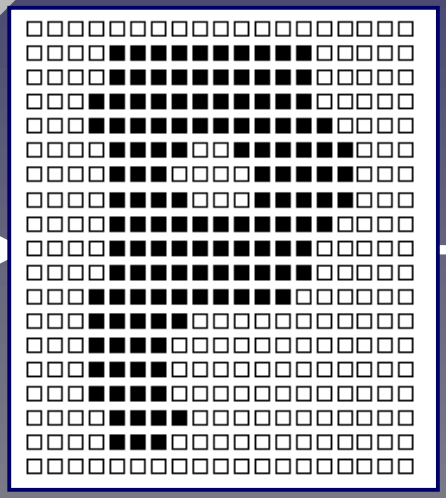
Segment Location



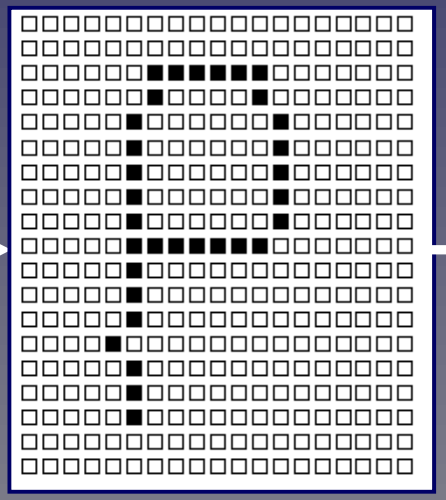
Isolated segment

Formatting the segments

Magnify

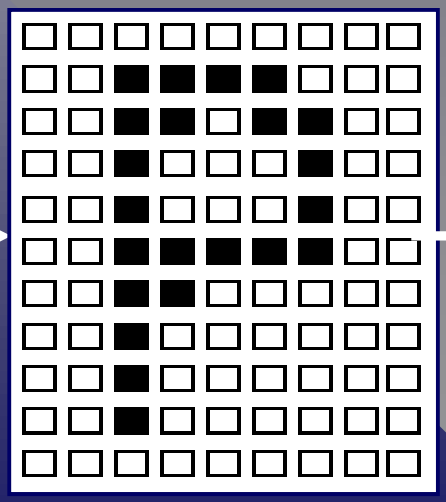


Biniarize Segment



Thin Segment

To Neocognitron



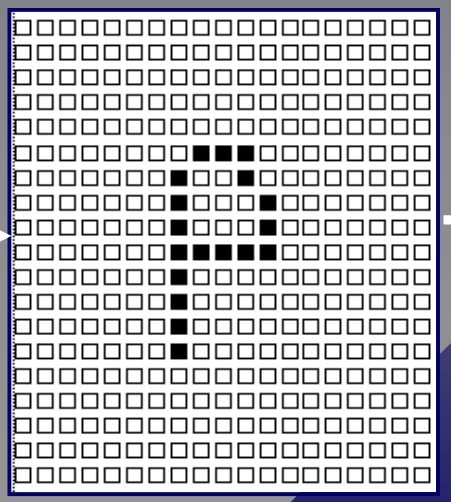


Image-processor's generality

Image-processor's generality

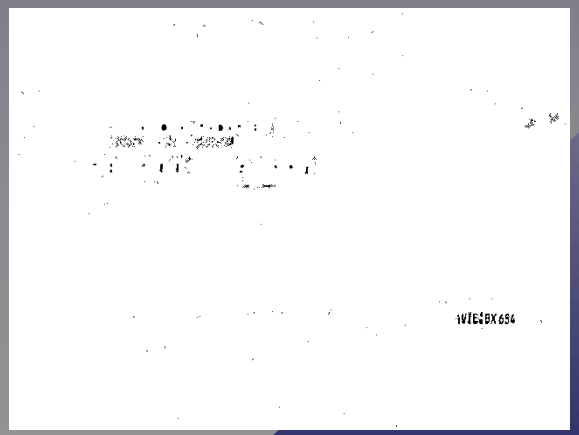
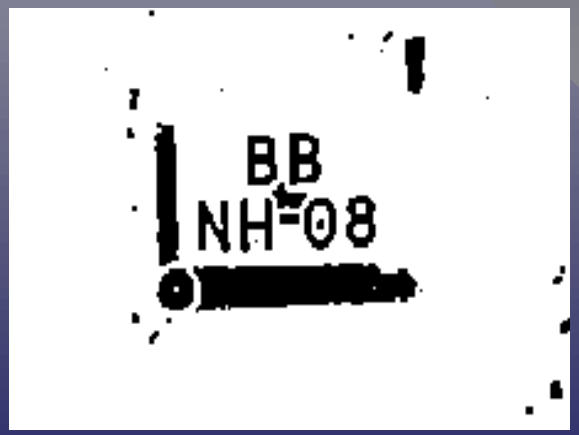
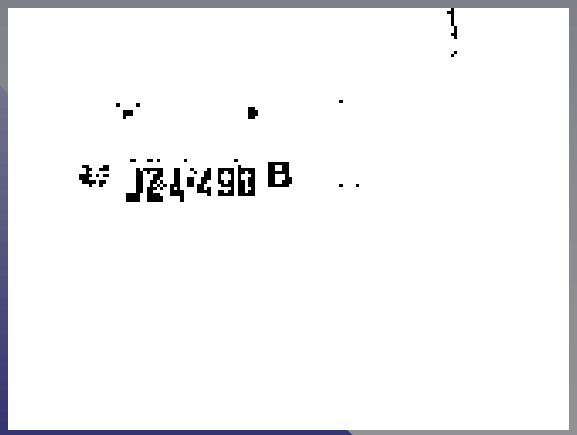
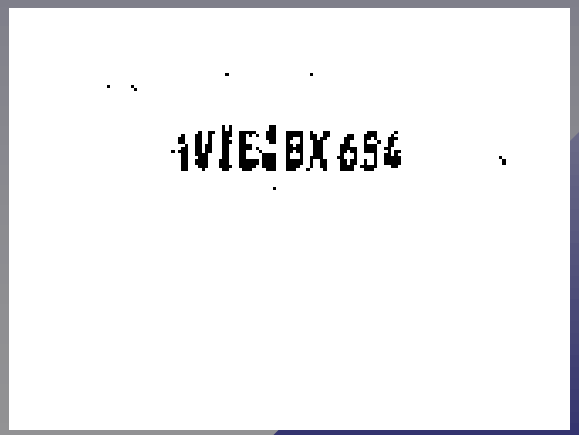


Image-processor's generality

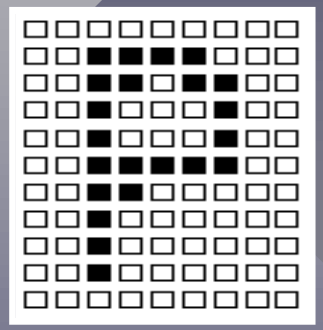






Character recognizer

- What: Classify the isolated image segments.



Direct input

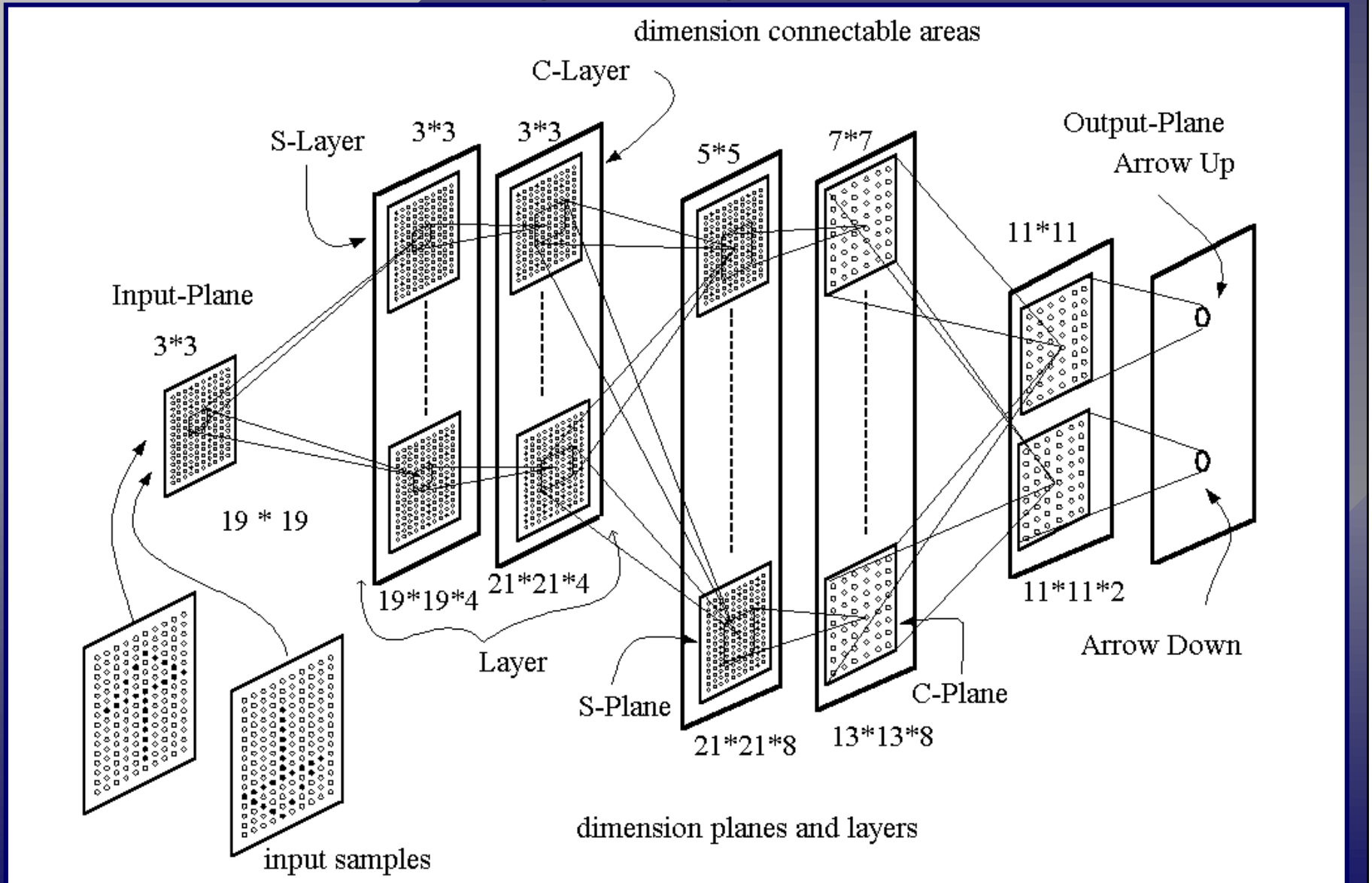
Output Classes

- How: Using a function from char-image → char-class

$$f: R^N \rightarrow R^M$$

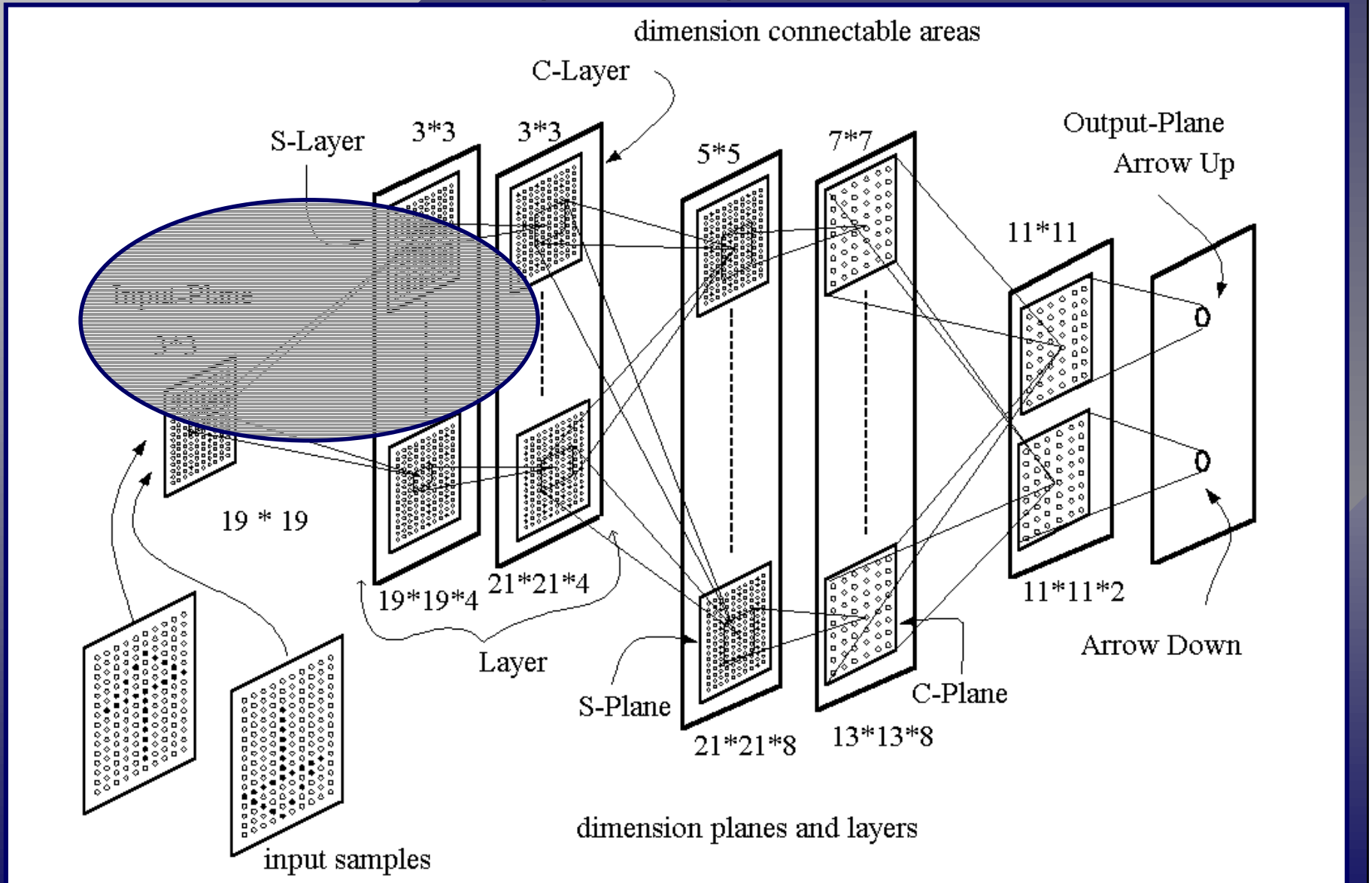
Neocognitron network

layout example.



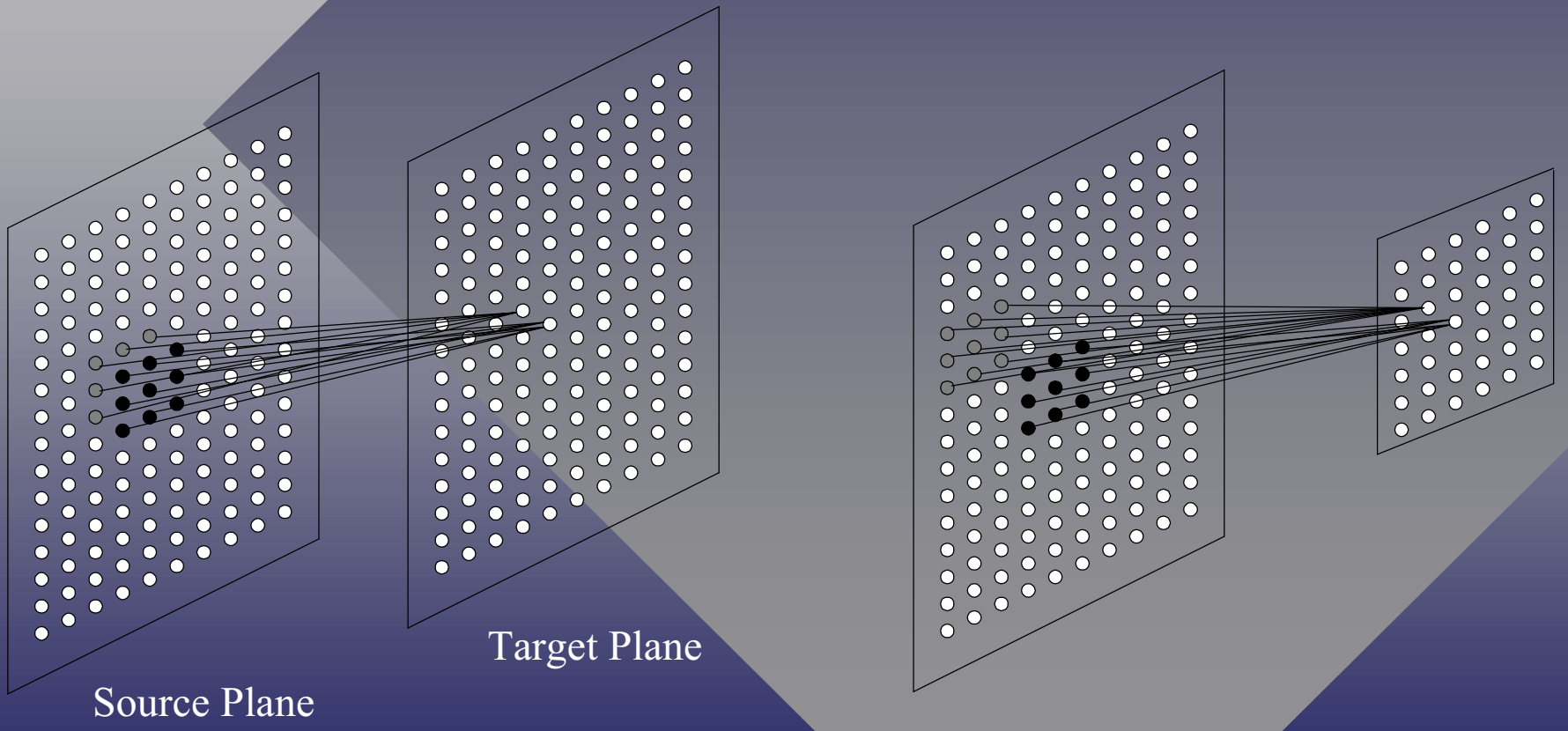
Neocognitron network

layout example.



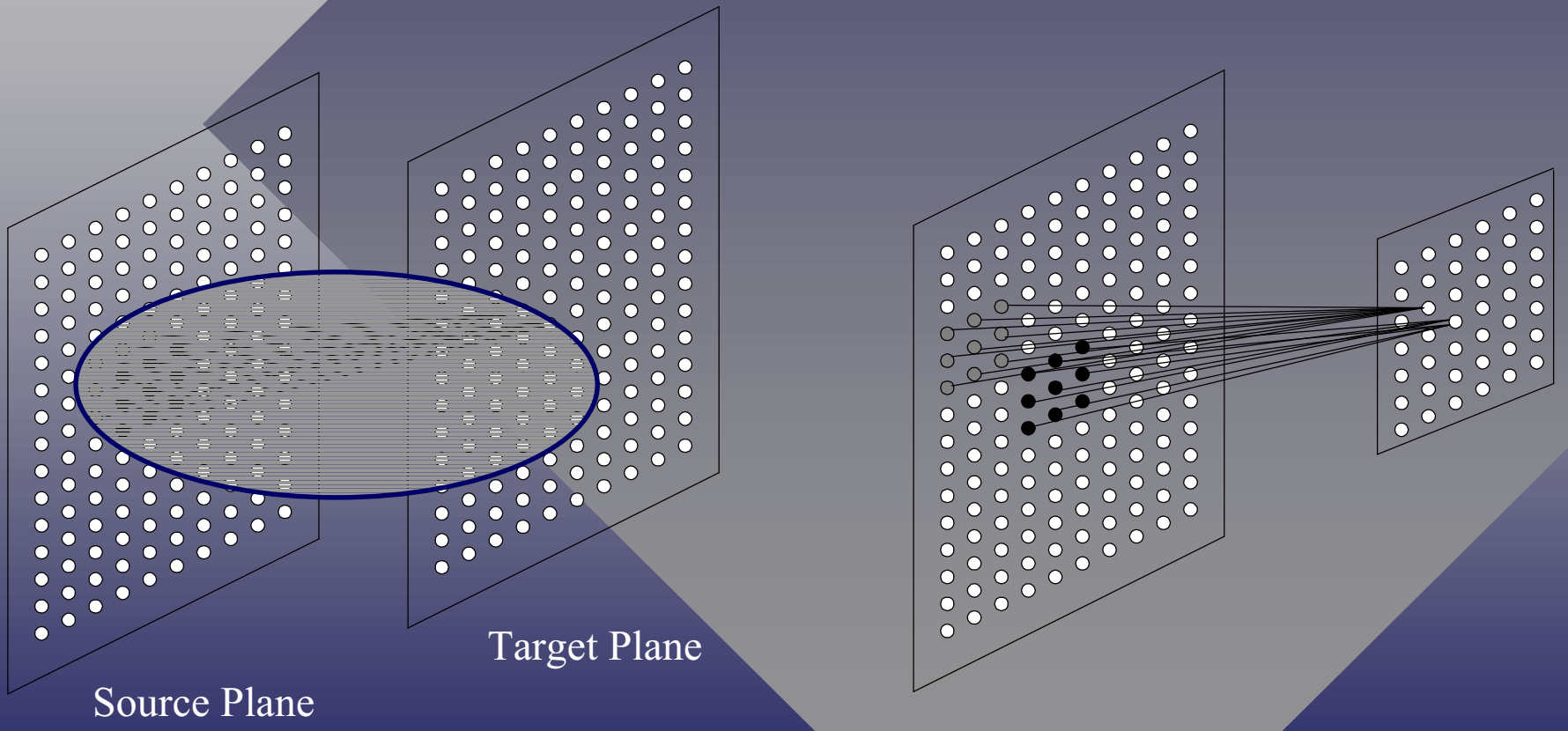
Neocognitron network

plane-connections.



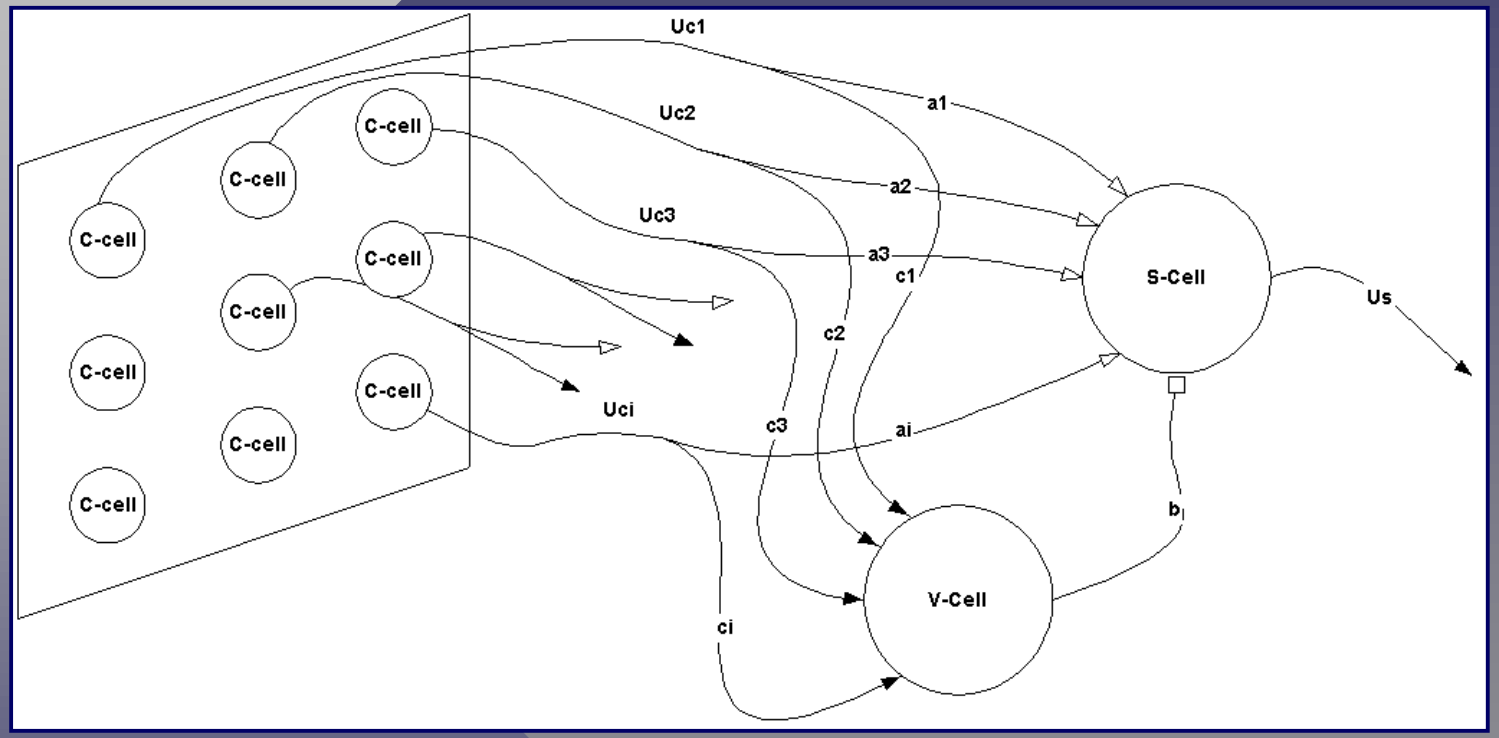
Neocognitron network

plane-connections.



Neocognitron network.

cell-connections.



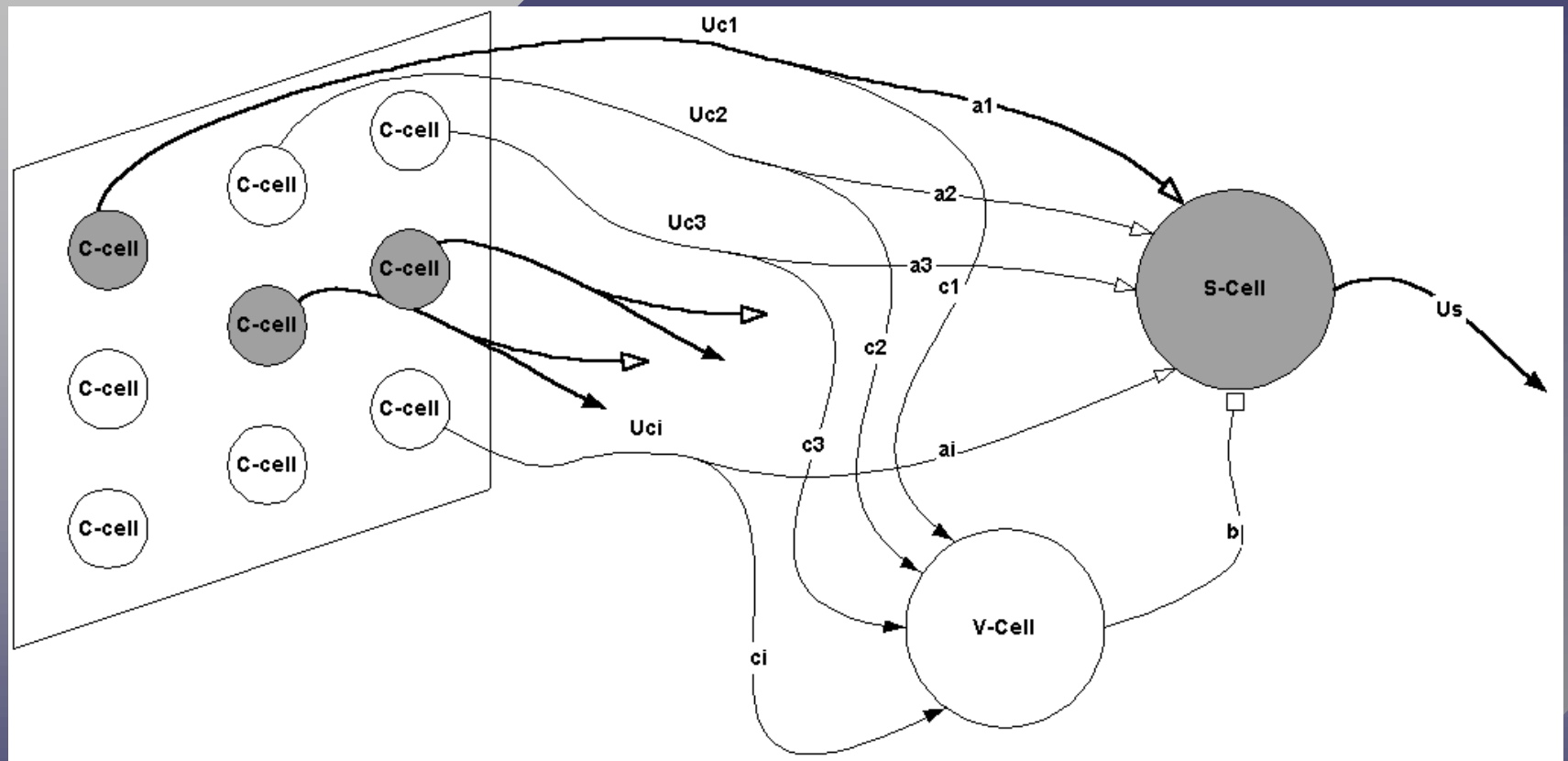
$$Uv = \sqrt{\sum_{i \in A} c_i U^2 c_i}$$

$$Us = r\varphi\left[\frac{1+\sum_{i \in A} a_i U c_i}{1+\frac{r}{r+1} bUv} - 1\right]$$

$$c_\lambda(v) = \gamma_\lambda^{|v|}$$

Neocognitron network.

cell-connections.



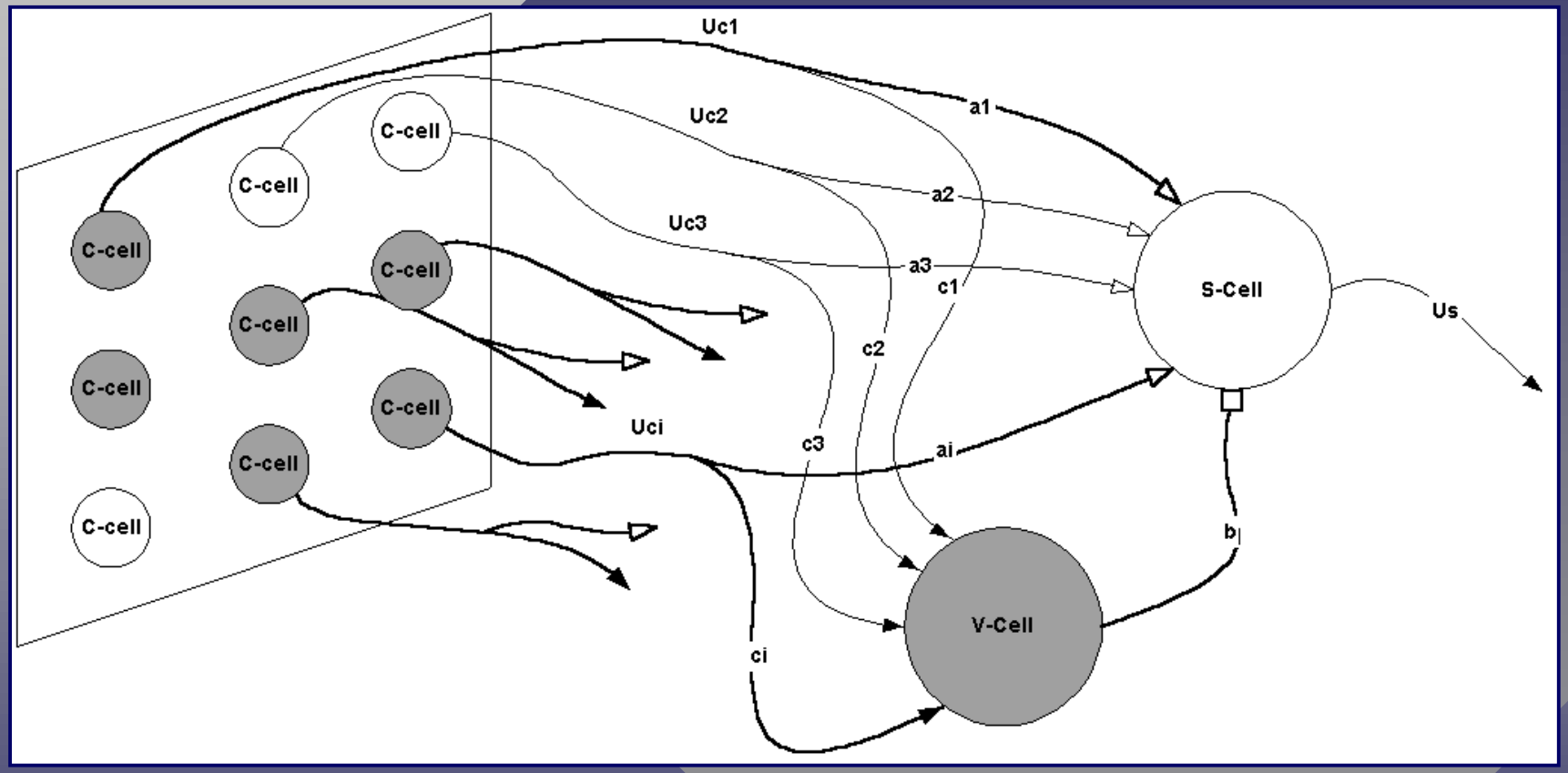
$$Uv = \sqrt{\sum_{i \in A} c_i U^2 c_i}$$

$$Us = r\varphi\left[\frac{1 + \sum_{i \in A} a_i U c_i}{1 + \frac{r}{r+1} bUv} - 1\right]$$

$$c_\lambda(v) = \gamma_\lambda^{|v|}$$

Neocognitron network.

cell-connections.



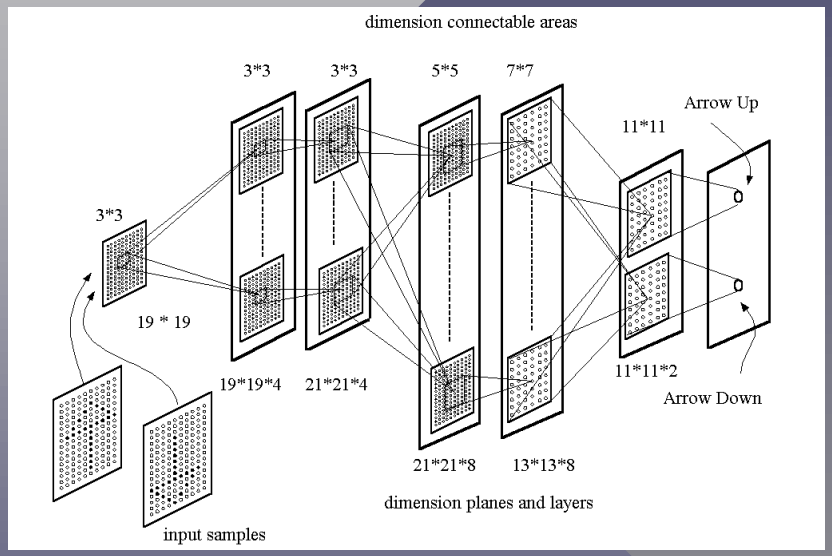
$$Uv = \sqrt{\sum_{i \in A} c_i U^2 c_i}$$

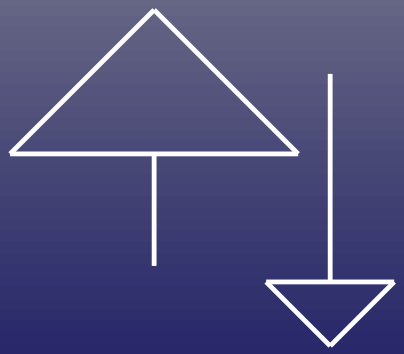
$$Us = r\varphi\left[\frac{1+\sum_{i \in A} a_i U c_i}{1+\frac{r}{r+1} bUv} - 1\right]$$

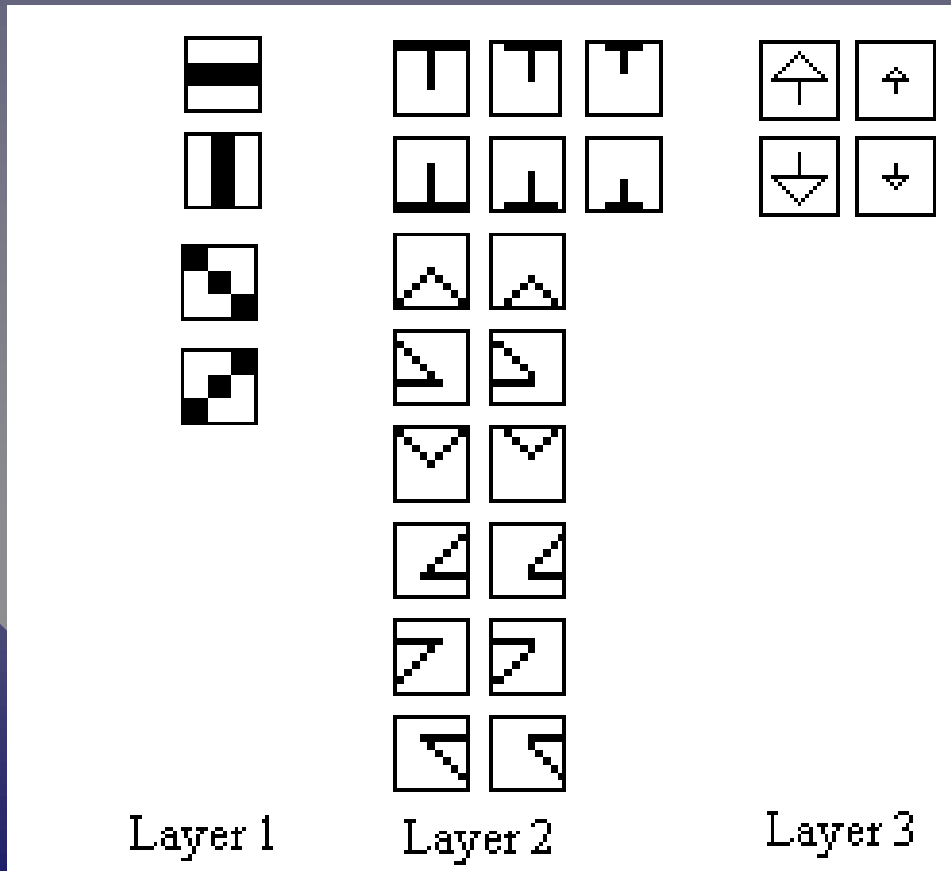
$$c_\lambda(v) = \gamma_\lambda^{|v|}$$

Neocognitron network.

training-set for recognition of *arrow* images.

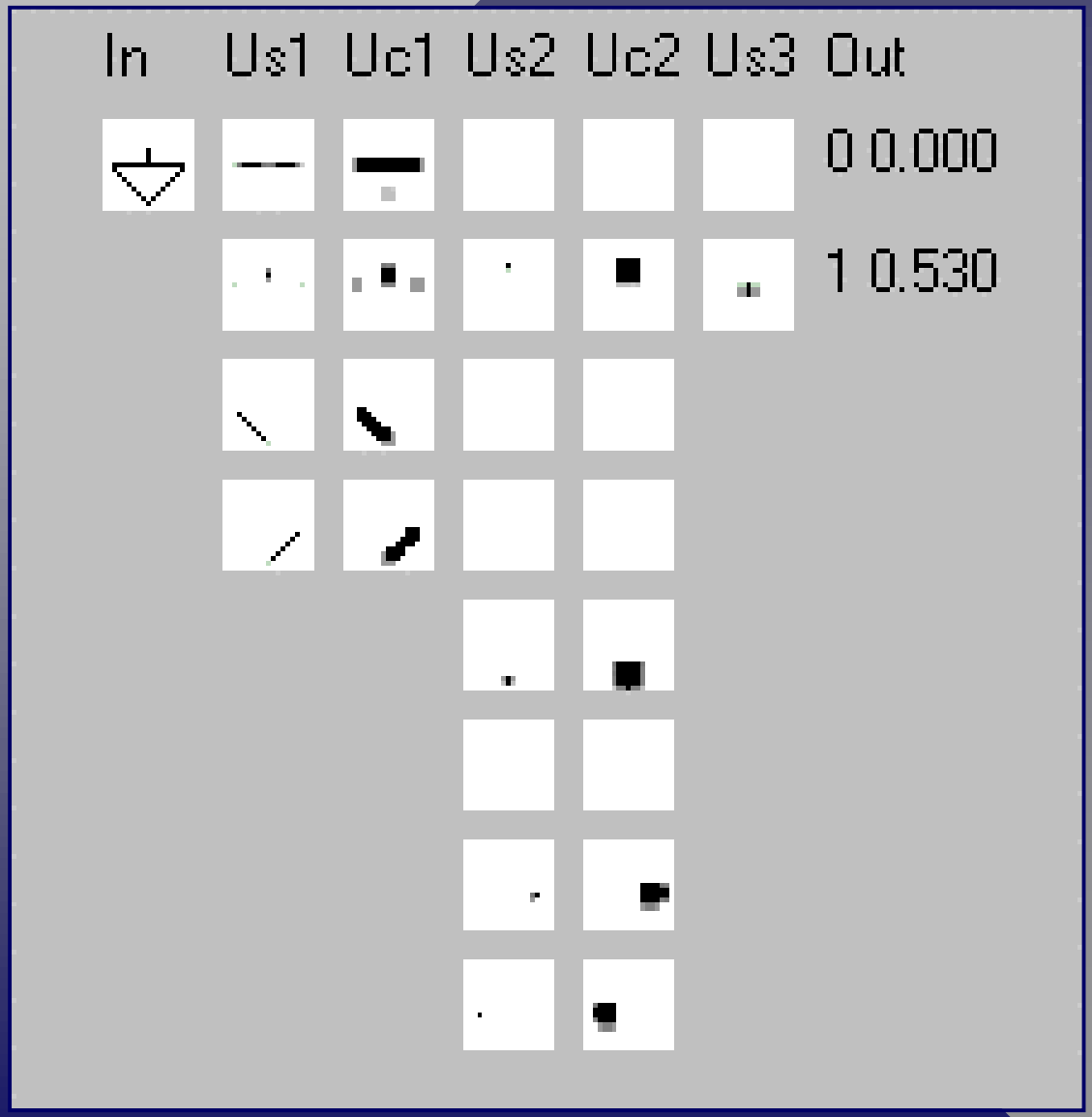


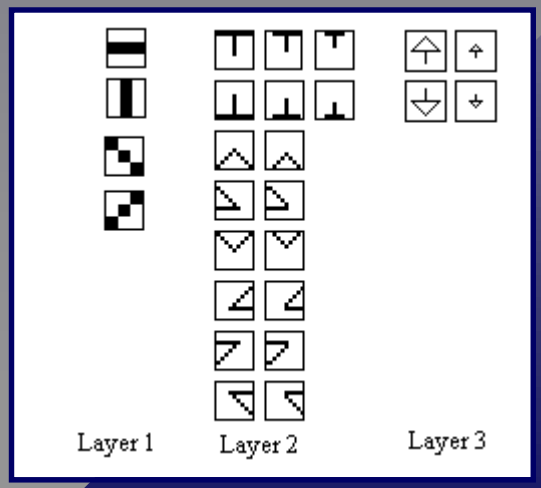




Neocognitron network.

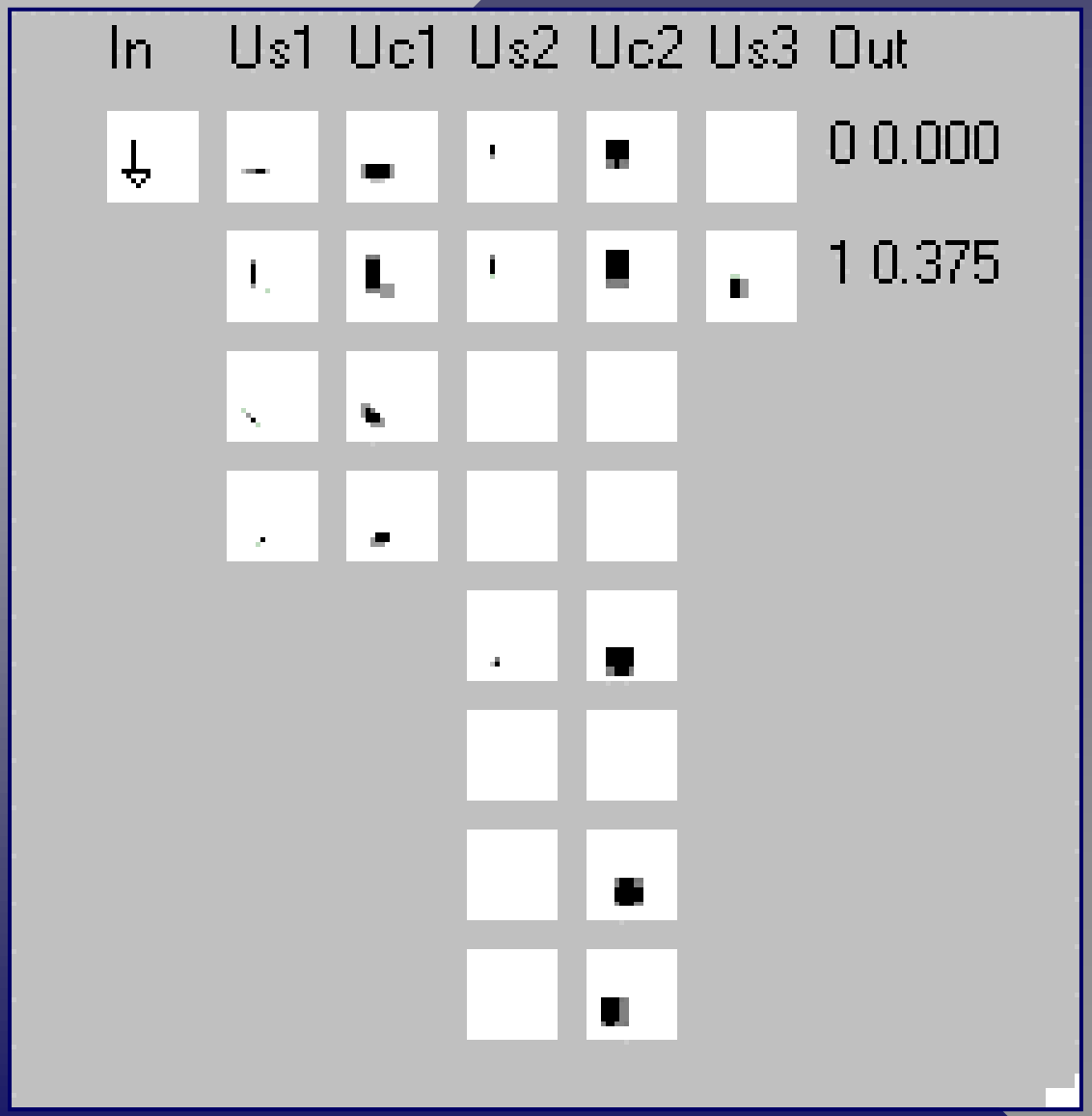
network response on *arrow* images.

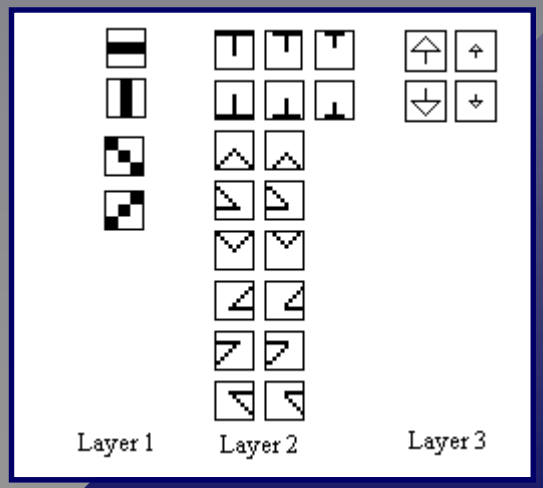




Neocognitron network.

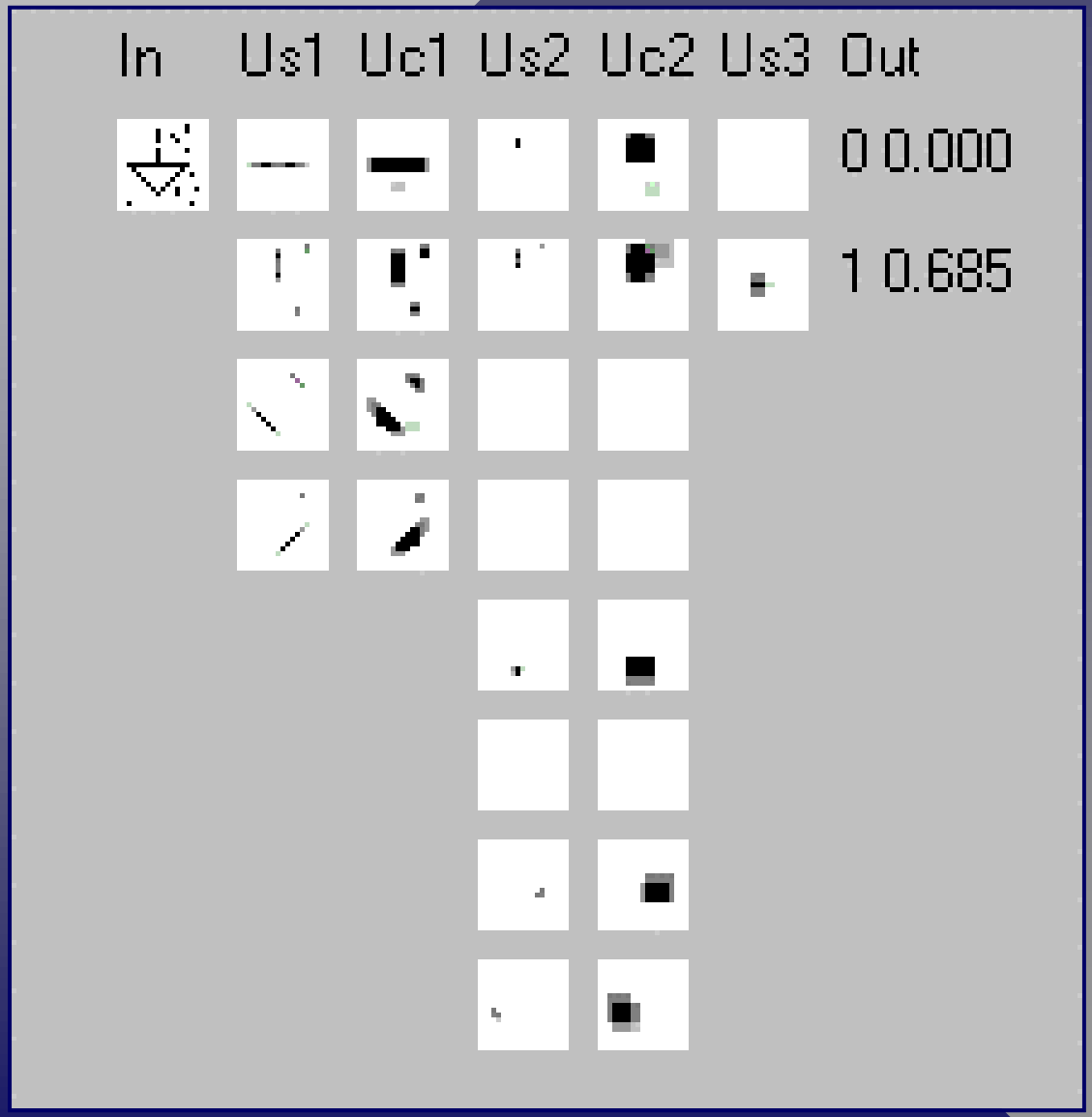
network response on *arrow* images.

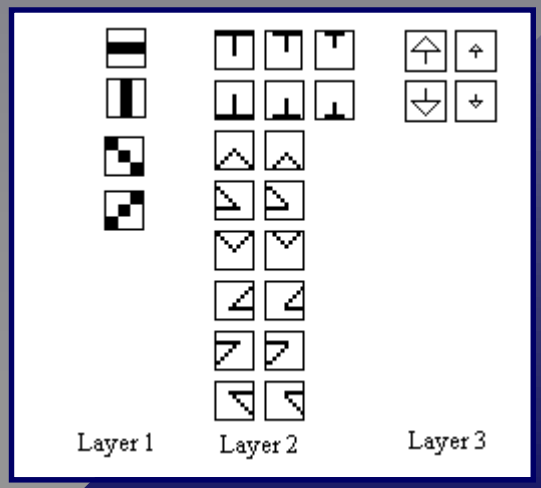




Neocognitron network.

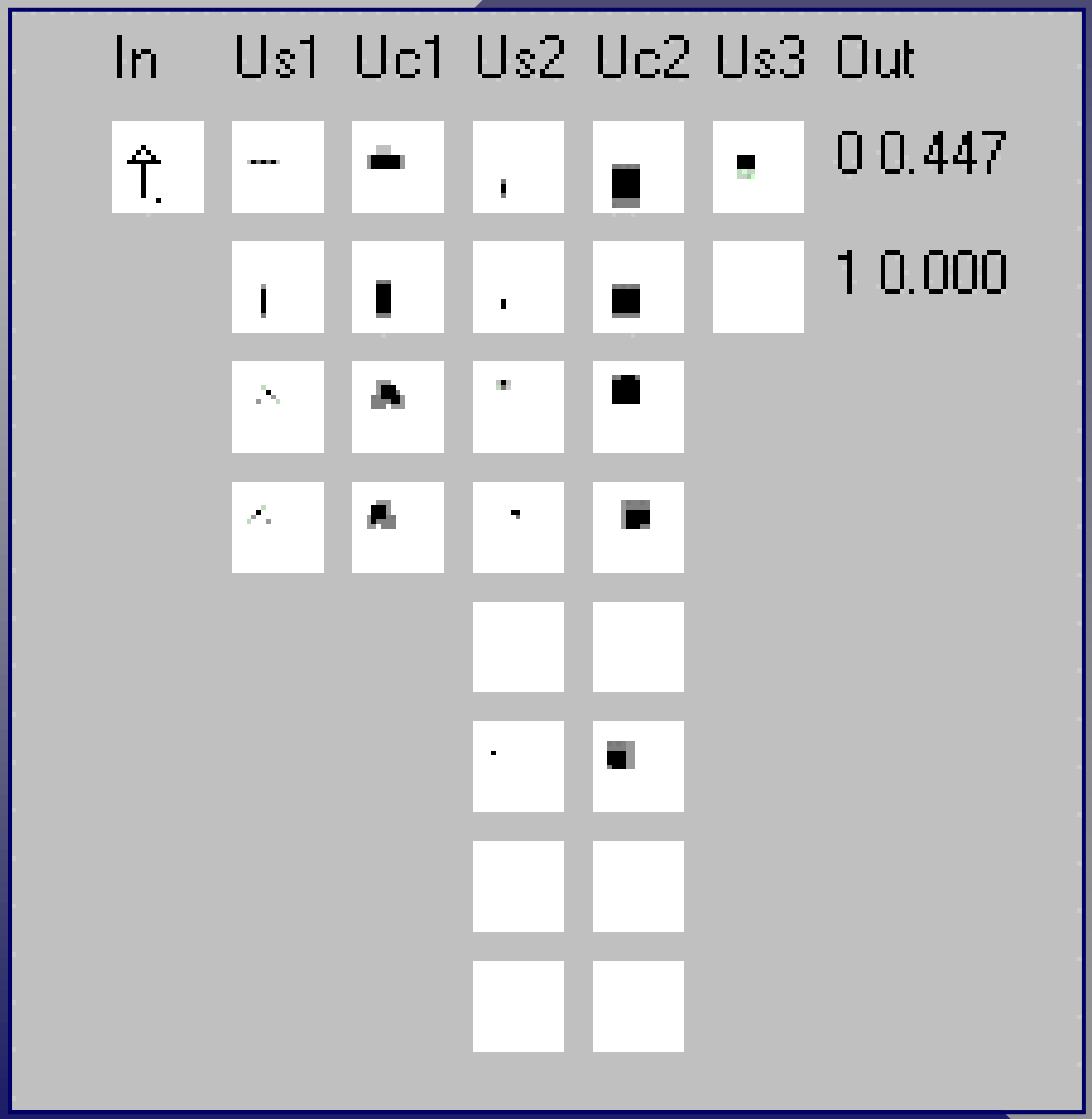
network response on *arrow* images.

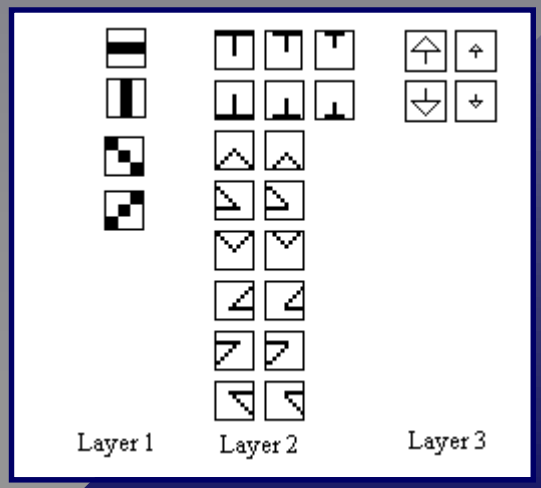




Neocognitron network.

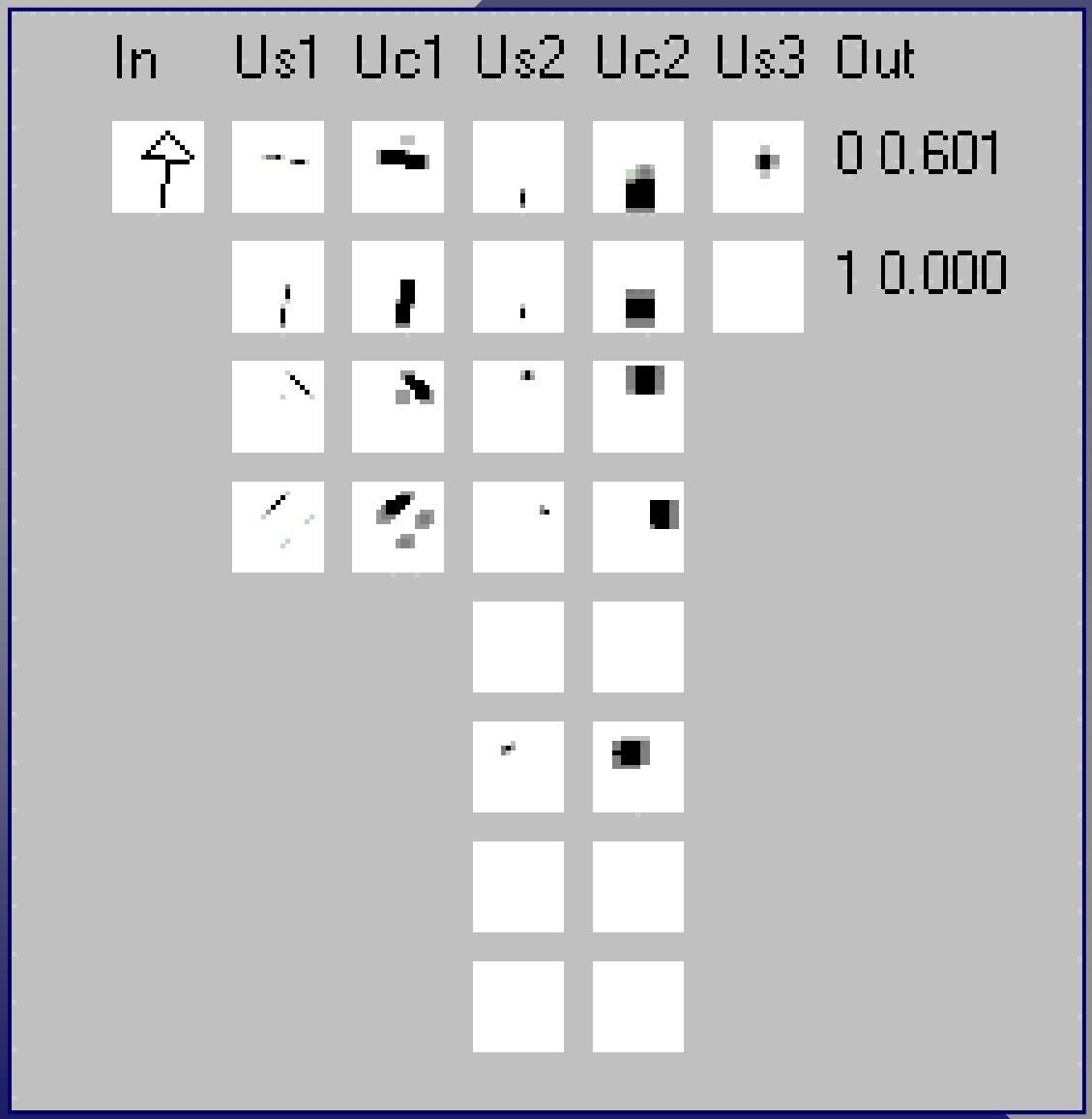
network response on *arrow* images.

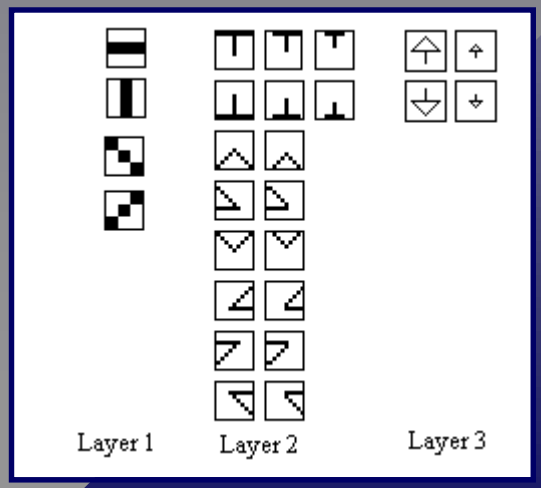




Neocognitron network.

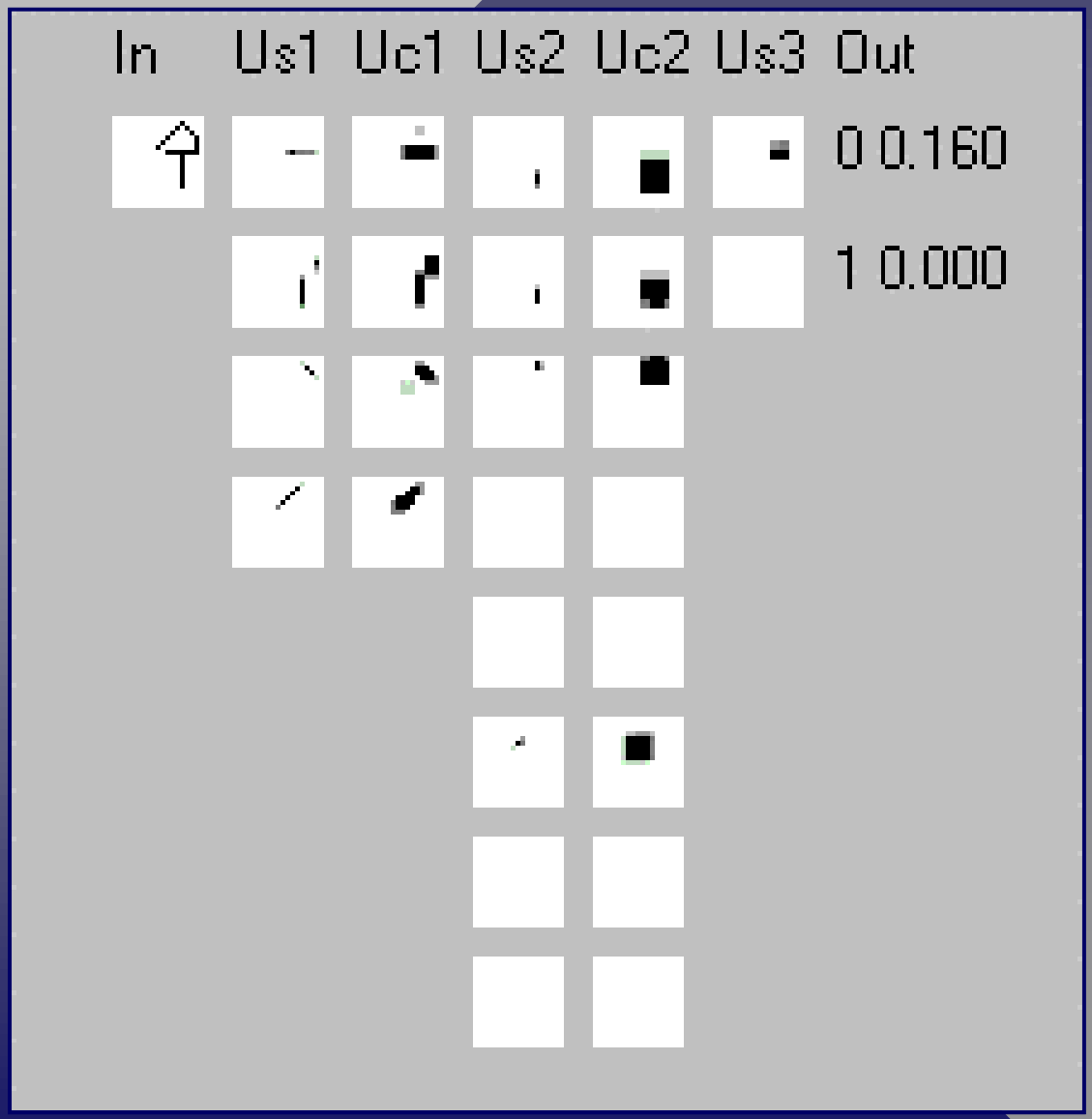
network response on *arrow* images.

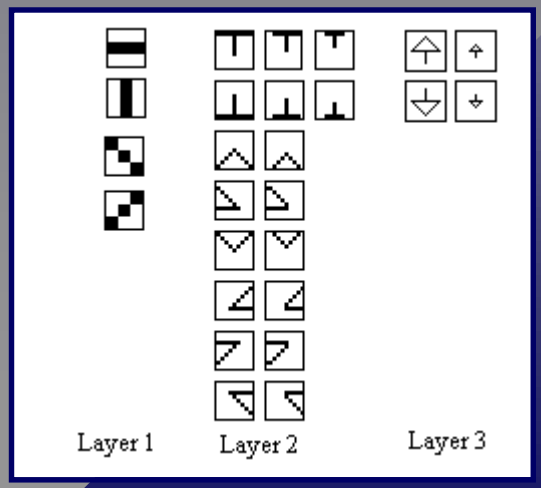




Neocognitron network.

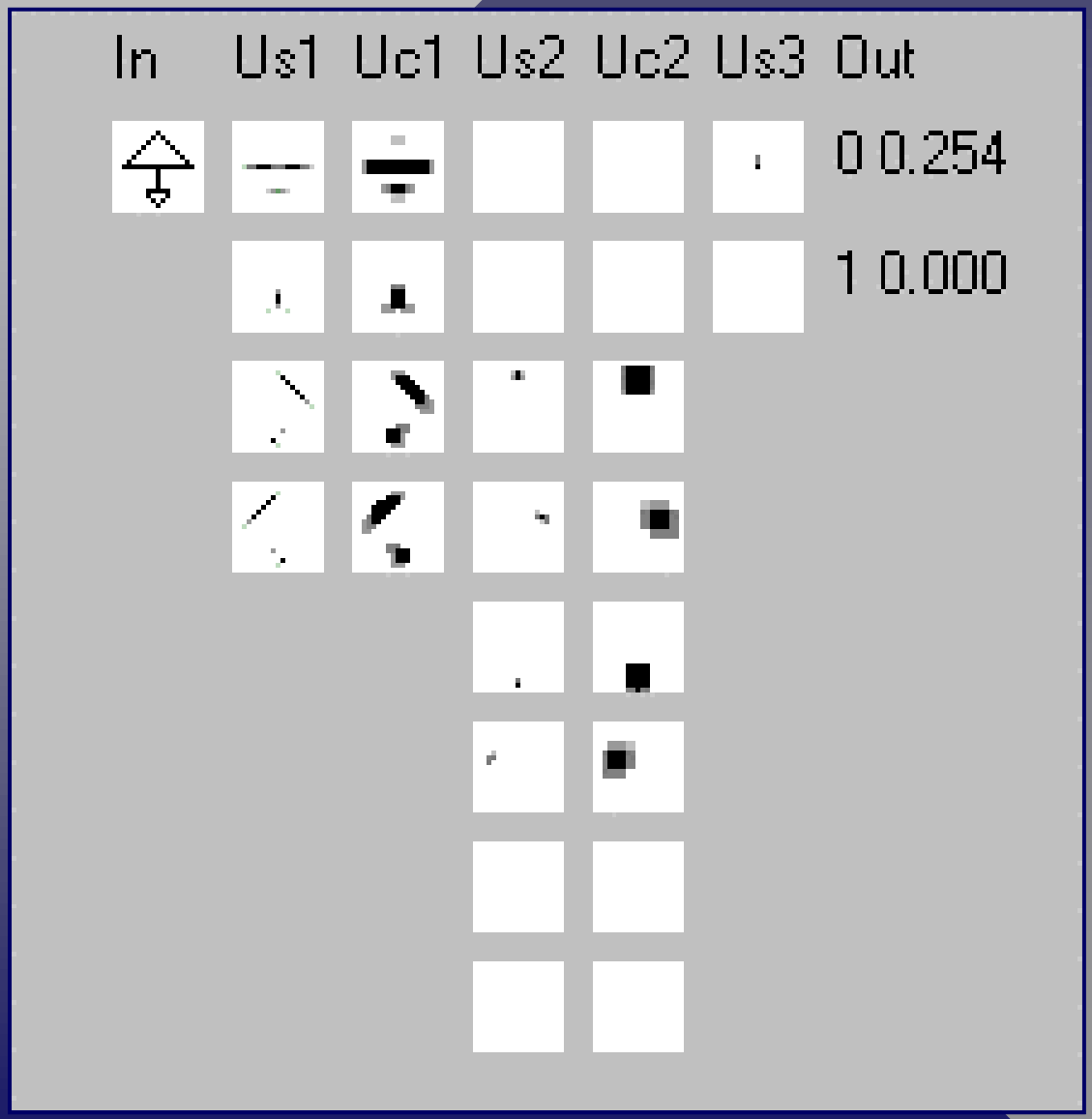
network response on *arrow* images.

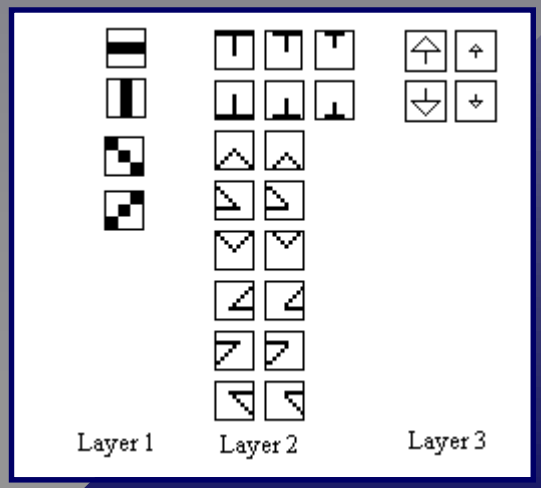




Neocognitron network.

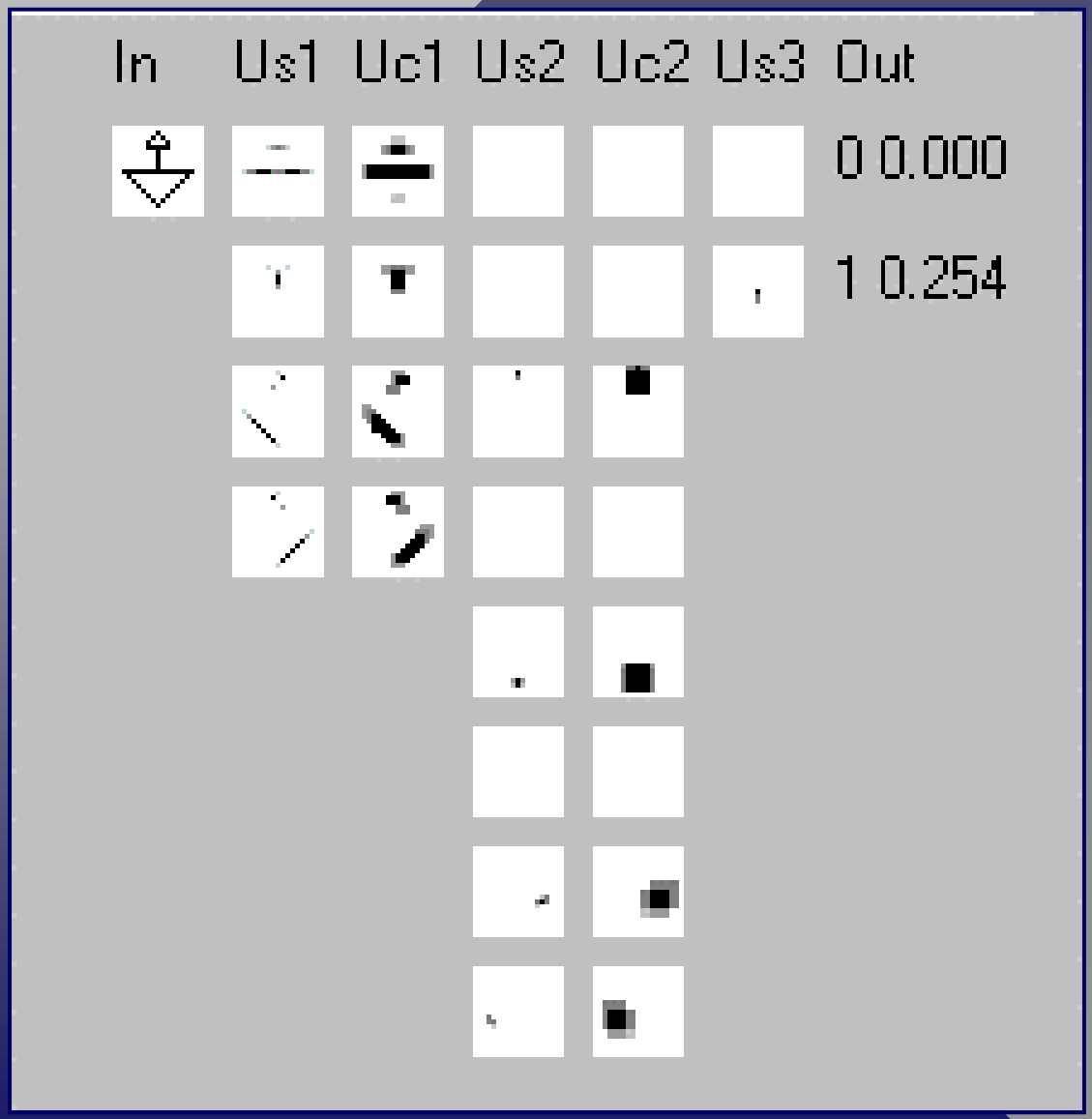
network response on *arrow* images.

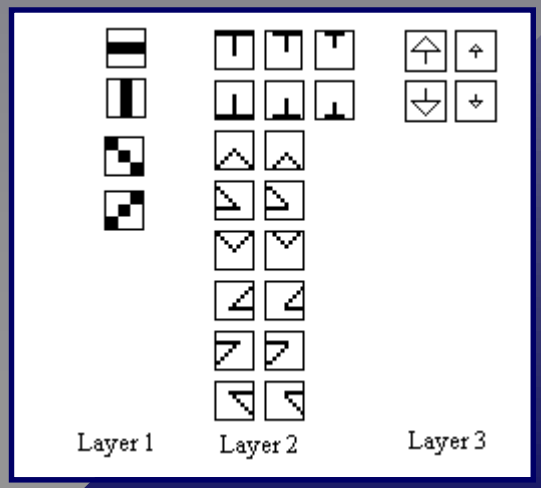




Neocognitron network.

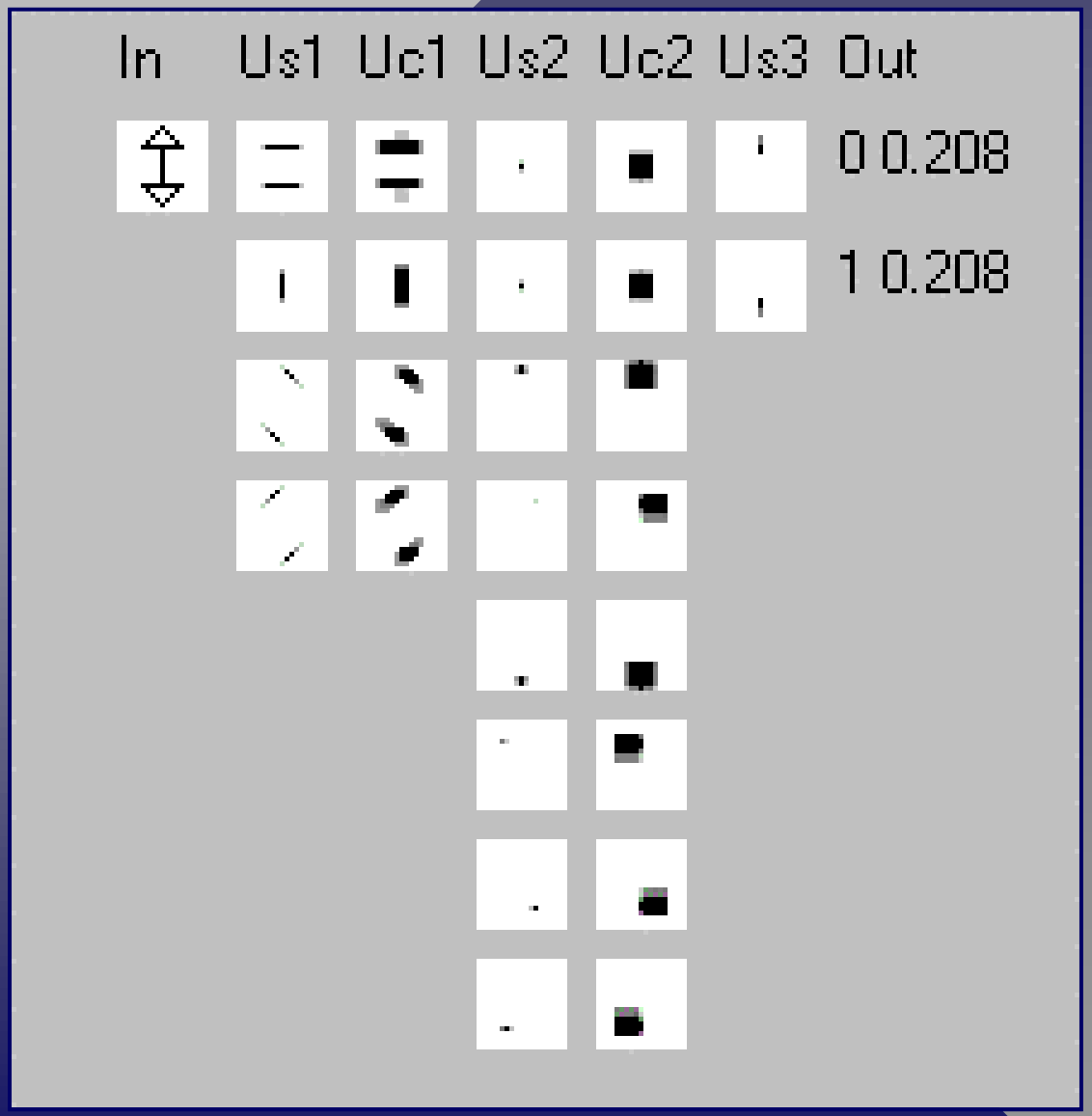
network response on *arrow* images.

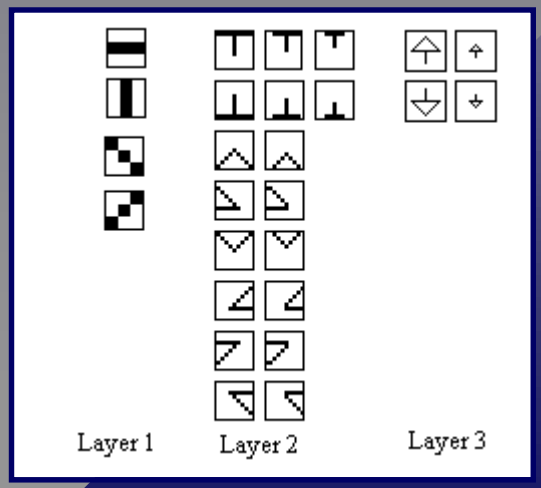




Neocognitron network.

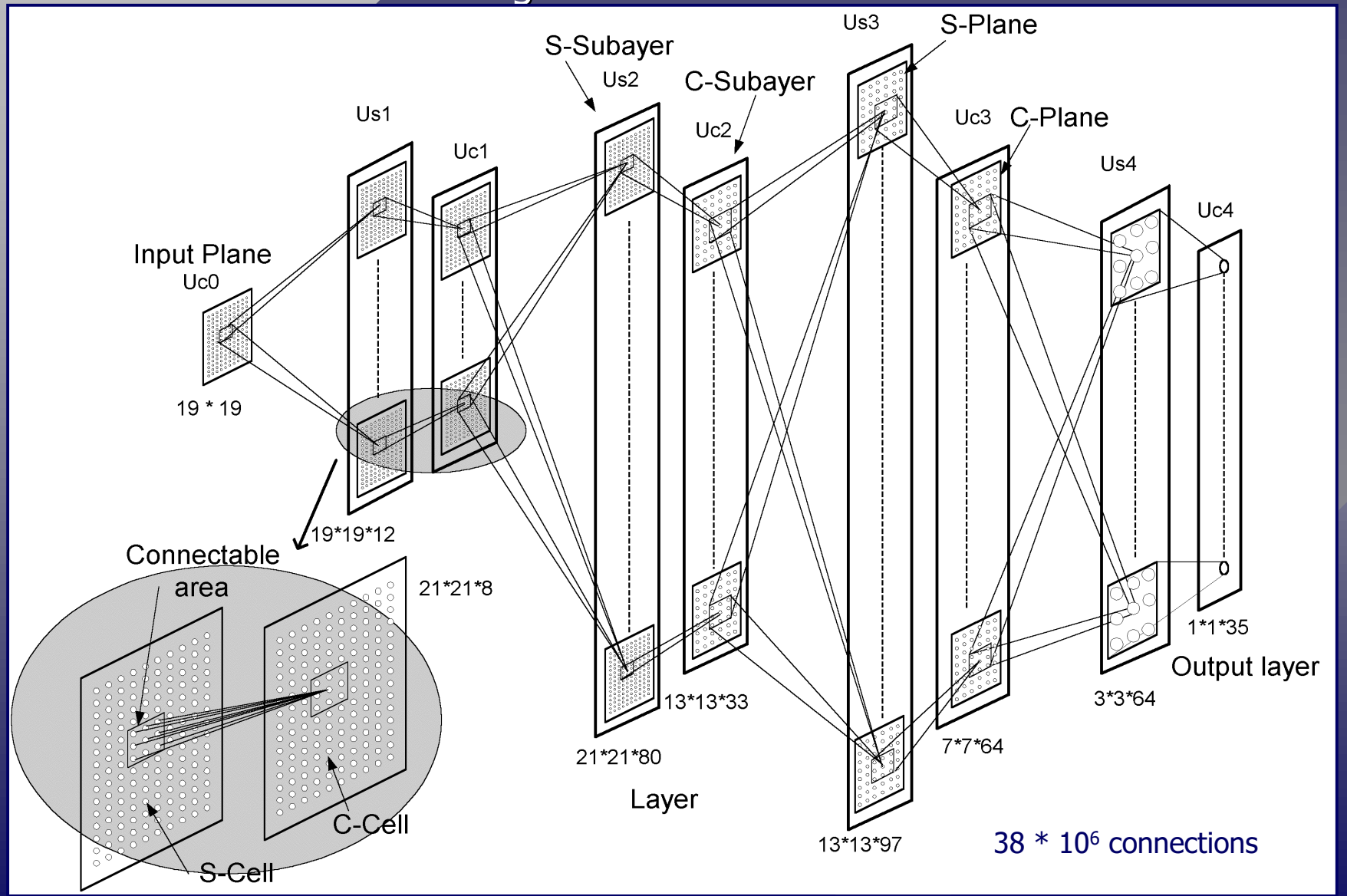
network response on *arrow* images.





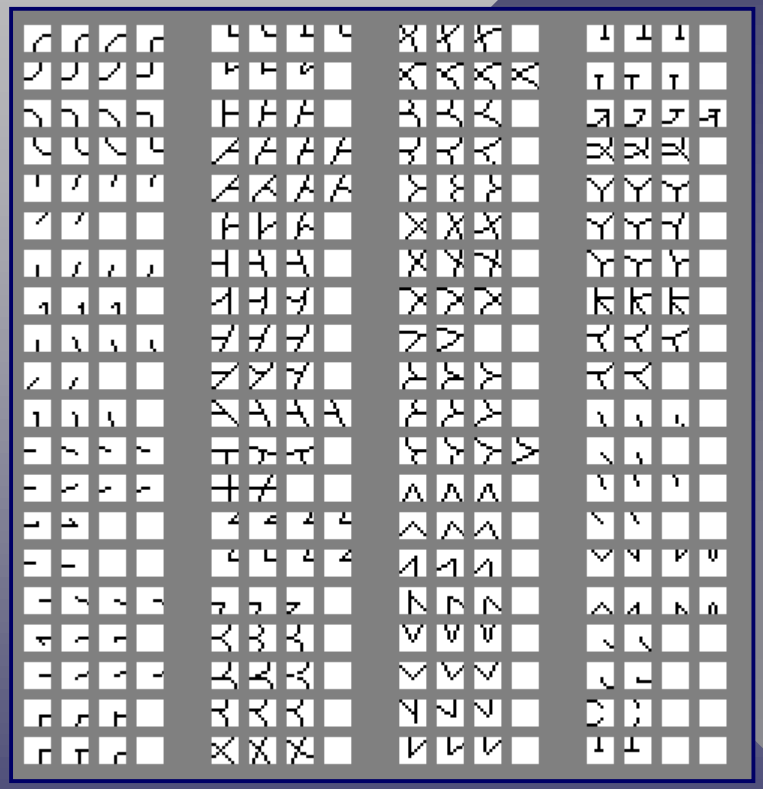
Neocognitron network.

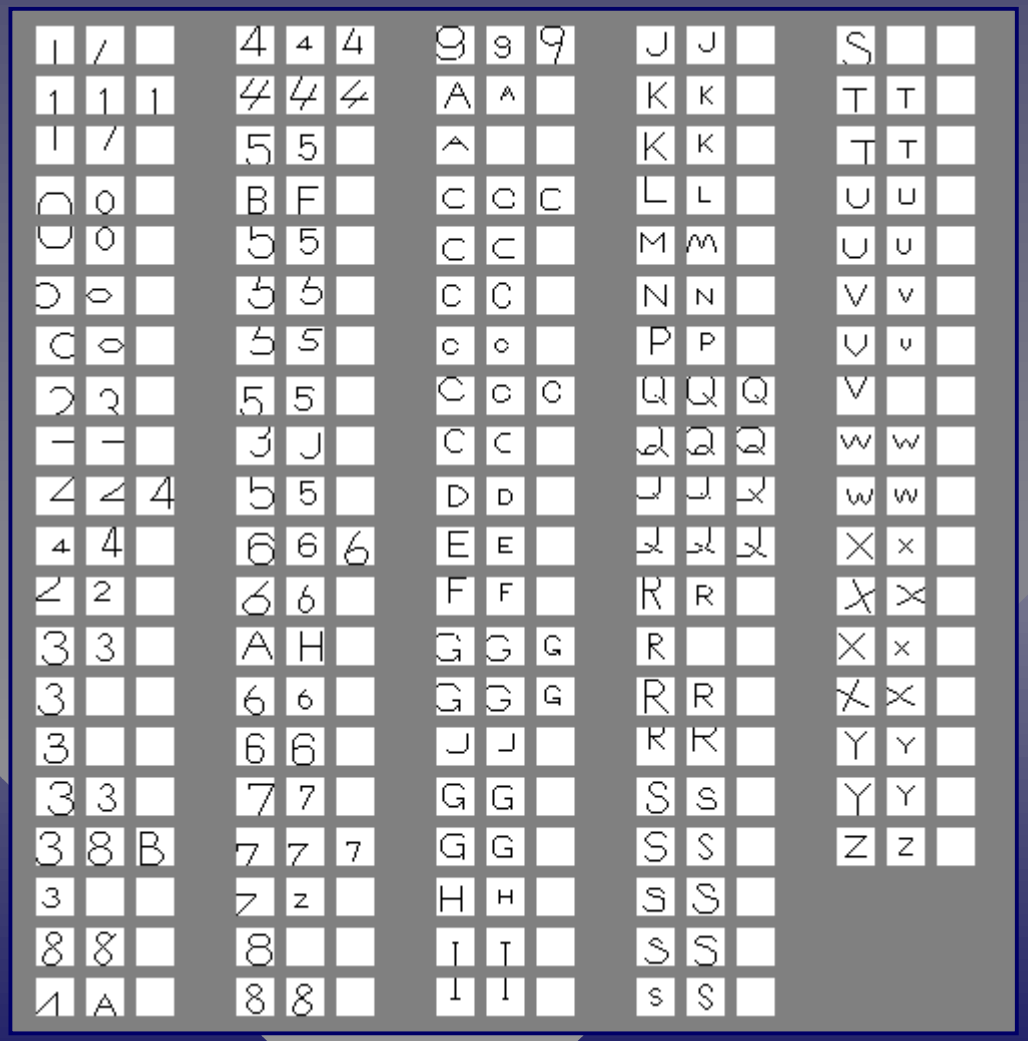
network to recognize handwritten characters.



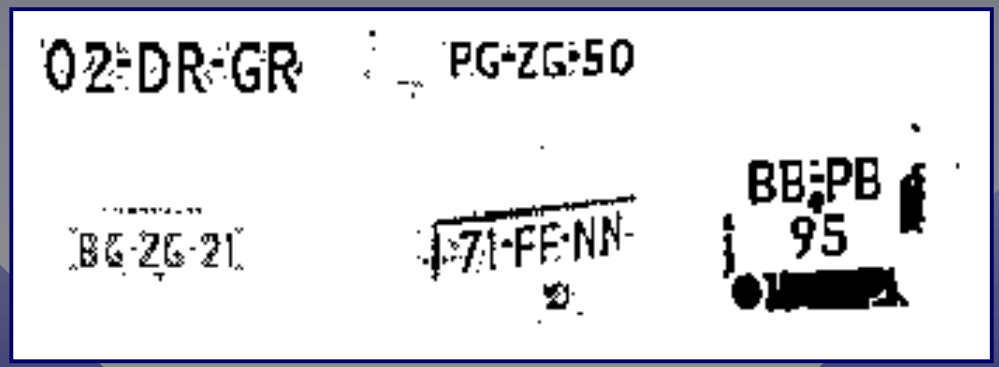
Neocognitron network.

Fukisuma's training set for layer 2 and layer 3





Application neocognitron



Recognition performance

- Network designed to recognize handwritten characters performs very poor on printed characters !

Handwritings are a generalisation of printed characters ?

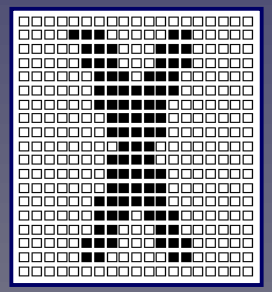
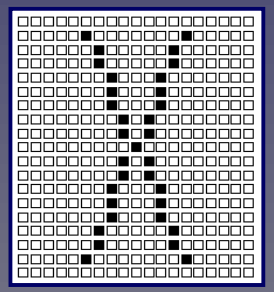
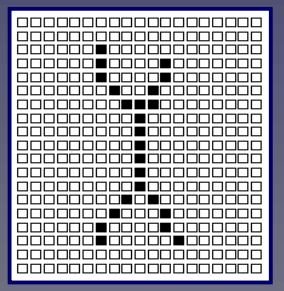
Poor recognition performance caused by ?

Recognizer problems

- Skeletonizer deformation
- Character style
- Ambigious characters

Recognizer problems

- Skeletonizer deformation

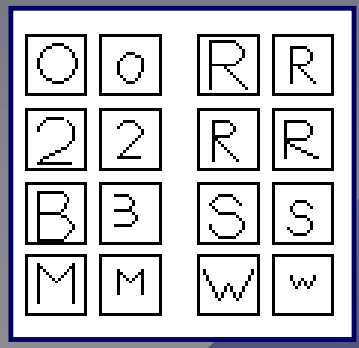




- Character style
- Ambigious characters

Recognizer problems

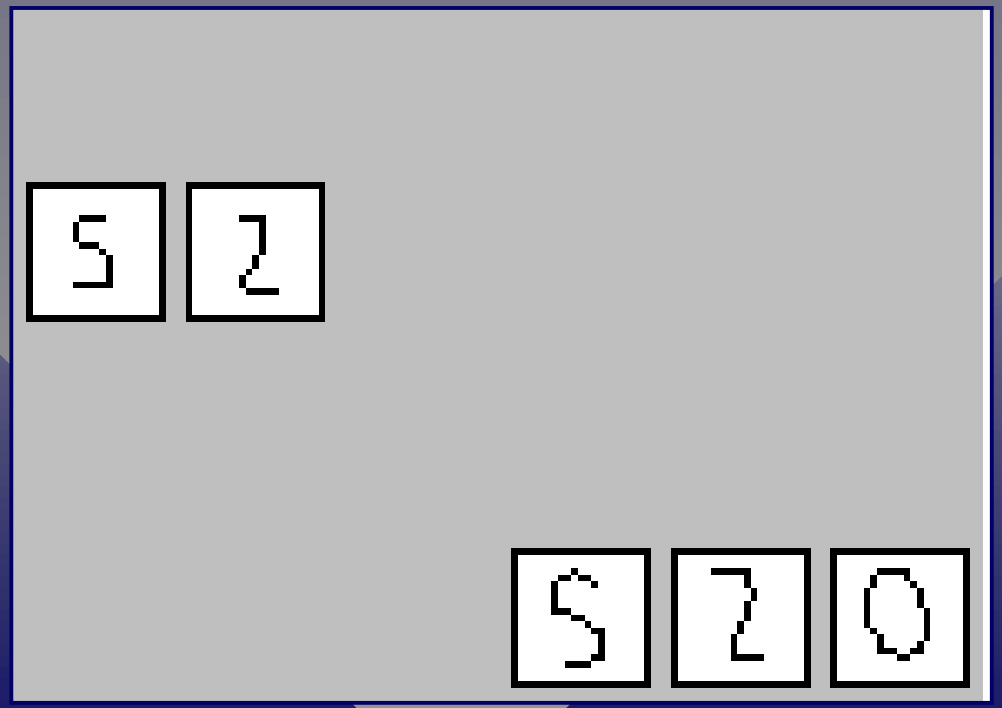
- Skeletonizer deformation
- Character style
- Ambigious characters

EuroStile font	0	2	B	M	R	S	W
Recognition rate	29%	29%	14%	0 %	14%	14%	29%



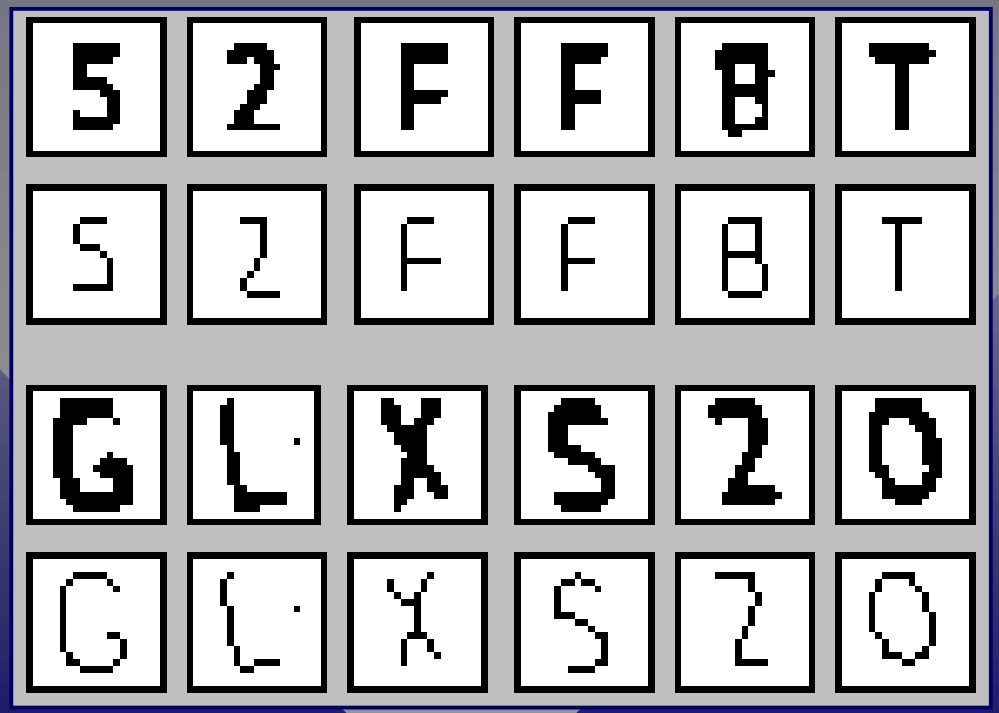
Recognizer problems

- Skeletonizer deformation
- Character style
- Ambigious characters



Recognizer problems

- Skeletonizer deformation
- Character style
- A ers



Reconfigure the network.

"The optimal scale of the network changes depending on the set of patterns to be recognized. Although it is difficult to show precisely how to choose the network scale parameters, we can offer a few guidelines"

"a process of constructing a good training set requires hard labor with an increase in the number of characters to be recognized"

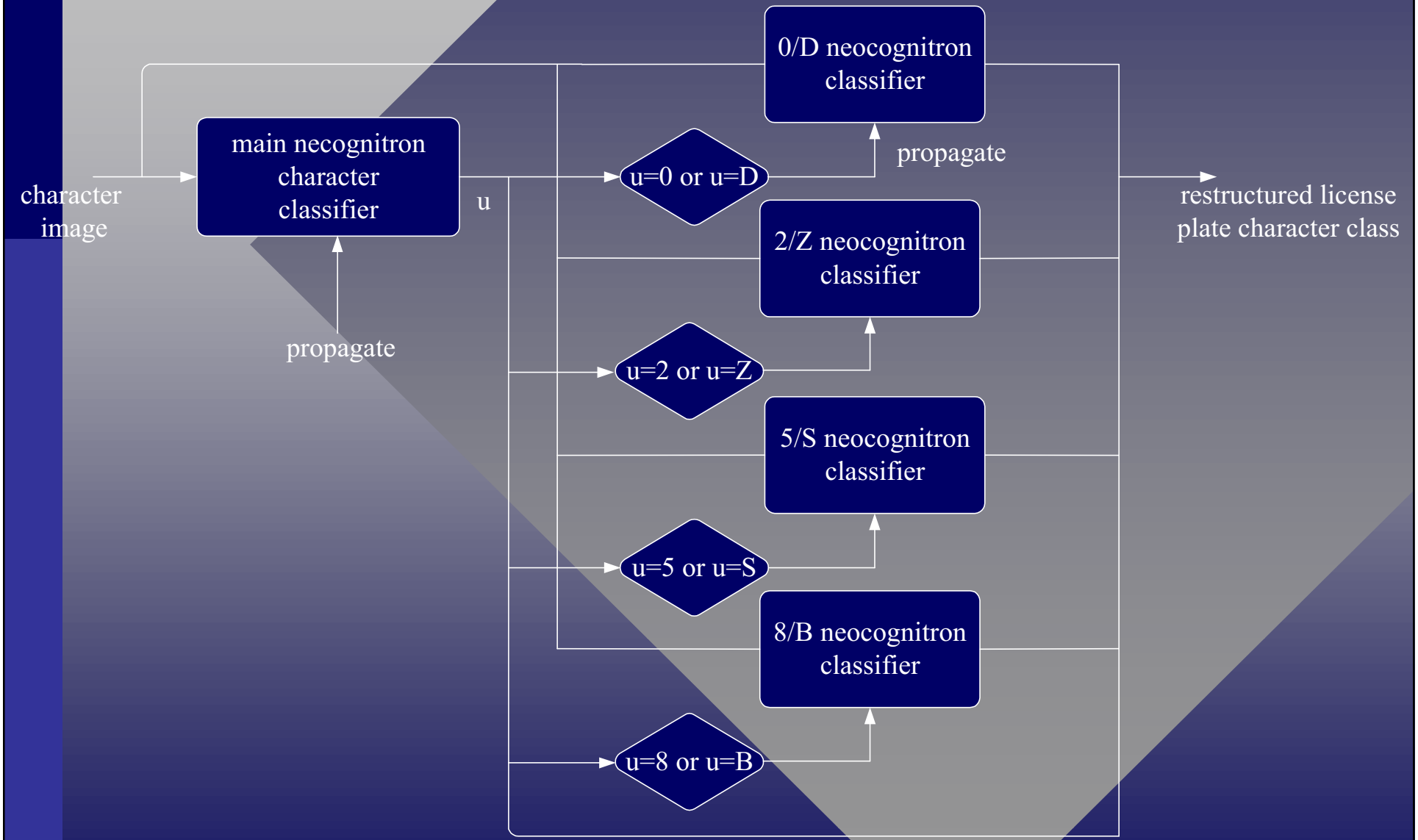
Fukisuma K. and Wake N. "Handwritten alphanumeric character recognition by the neocognitron", (1991) *IEEE Transactions on neural networks VOL 2. No 3. May 1991*, pp.355-365.

Neocognitron reconfiguring.

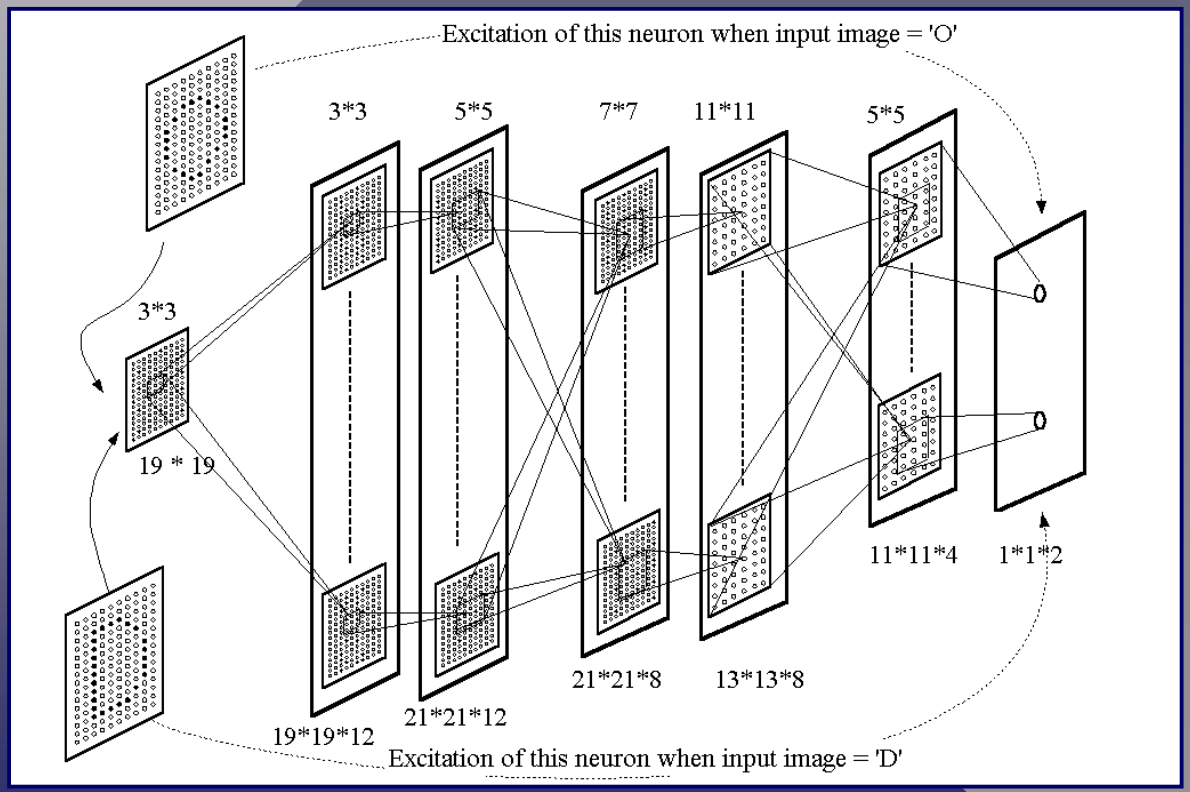
- New 3-layer network configuration & new training set patterns to solve skeletonizer deformation and character style incompatibilty.
  - Only license plates characters classes
  - Drop location invariance
  - Limitations on size variations

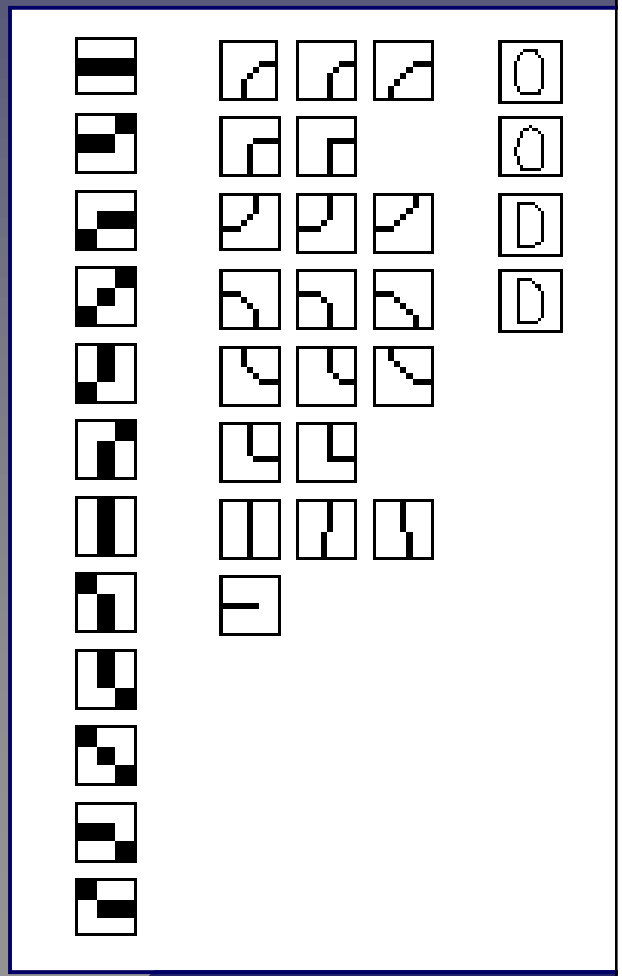
- Specific 3-layer networks for discrimination between ambigous characters 0/D, 2/Z, 5/S & 8/B.
  - avoid tuning a large network
  - small in size, short training times, easy to tune

Neocognitron specific classifiers

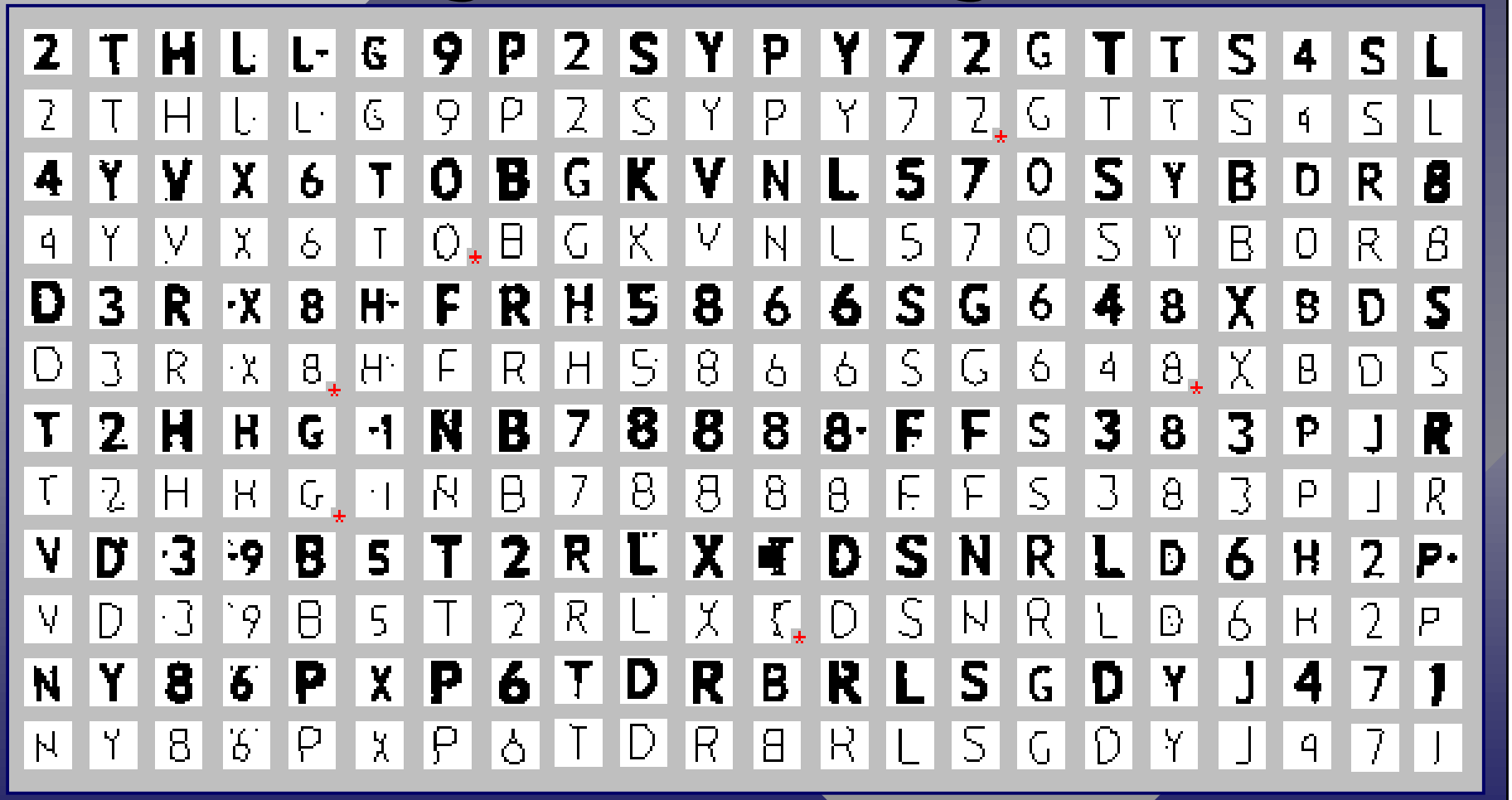


Neocognitron specific 0/D classifier





Classified correctly by the reconfigured neocognitron



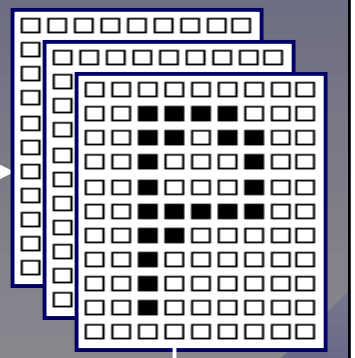
Character level 94 % recognition

CLPR conceptual diagram

image processor

JS-PI-84

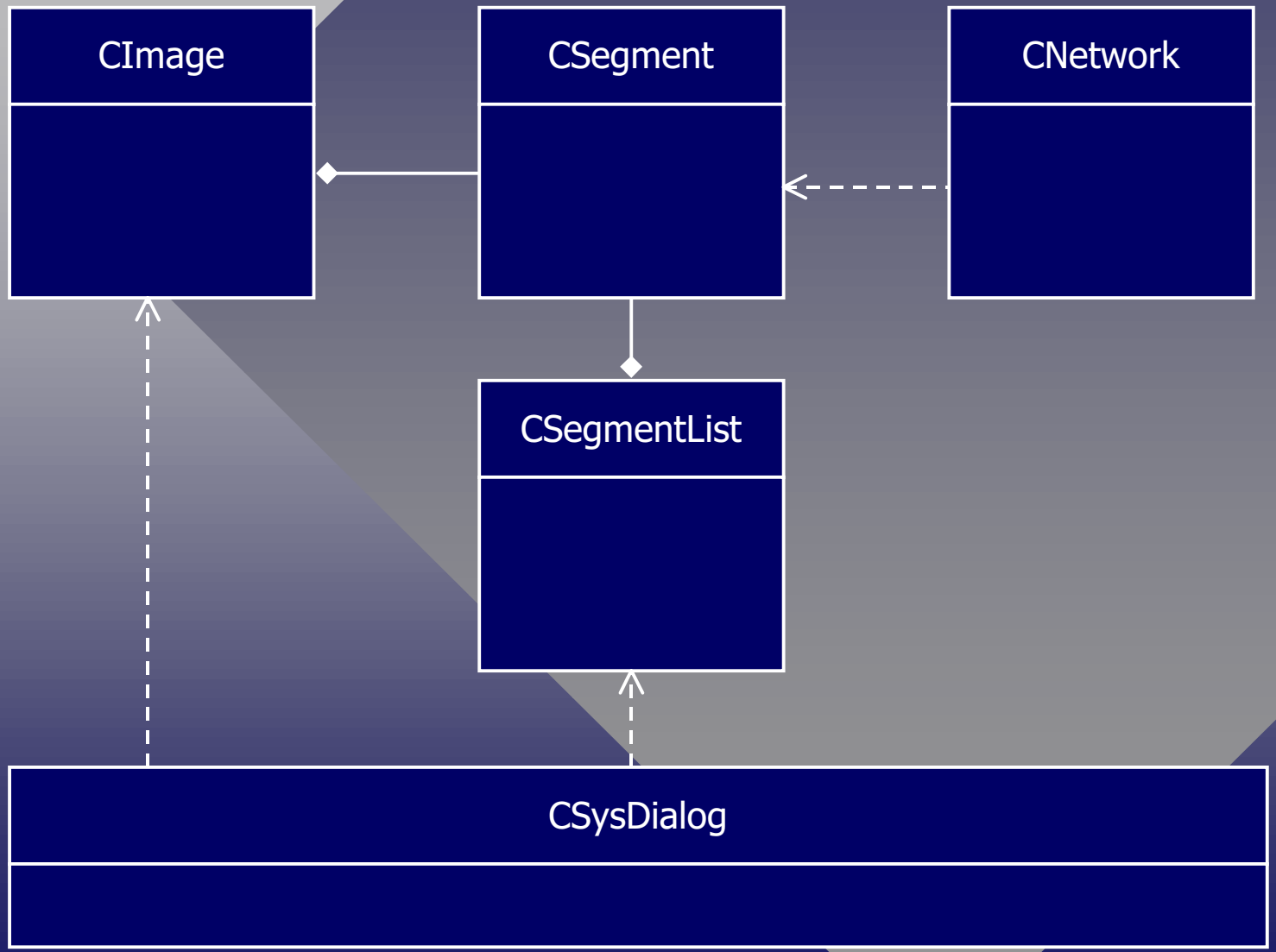
segment processor



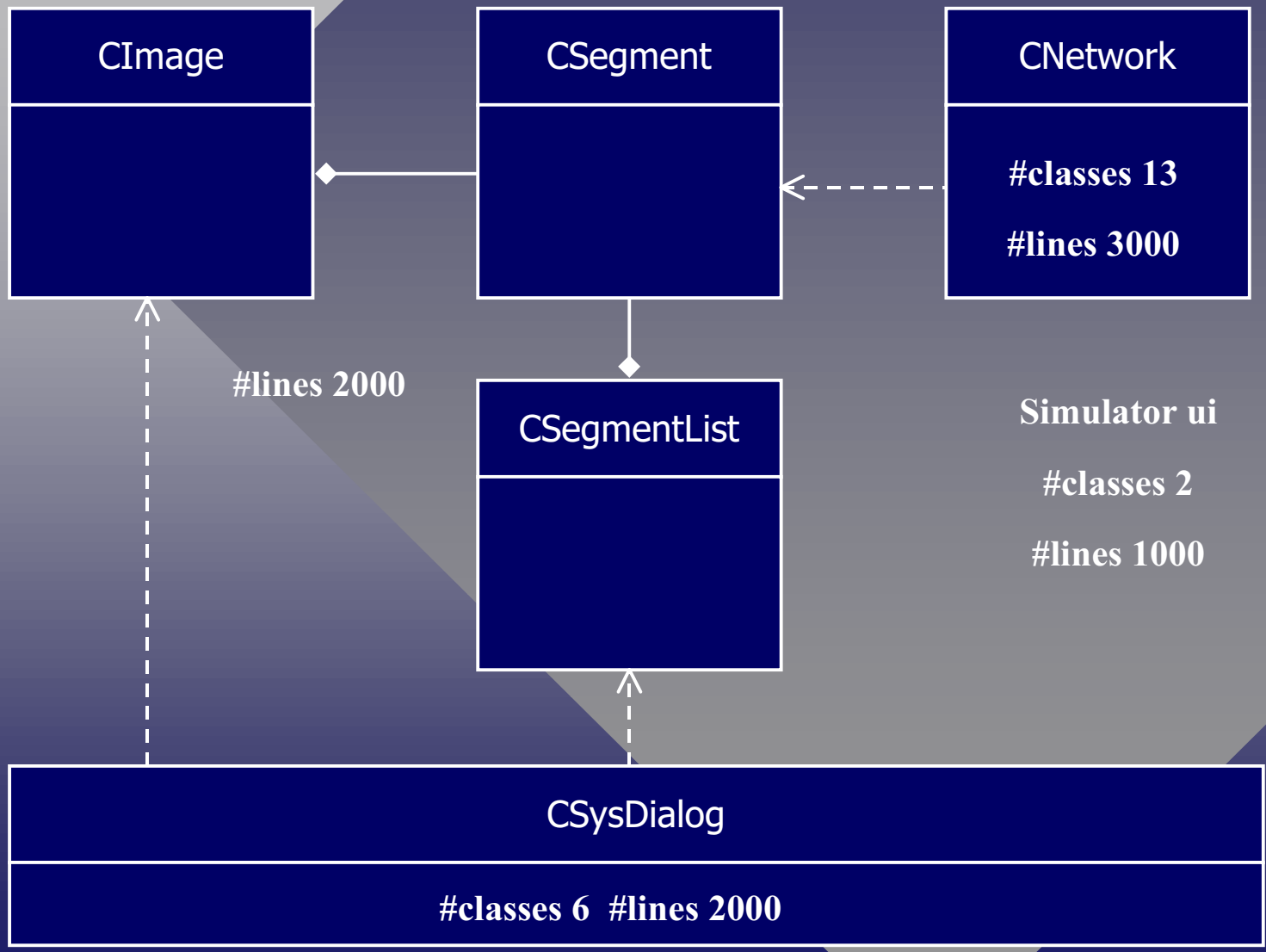
neocognitron simulator

User interface

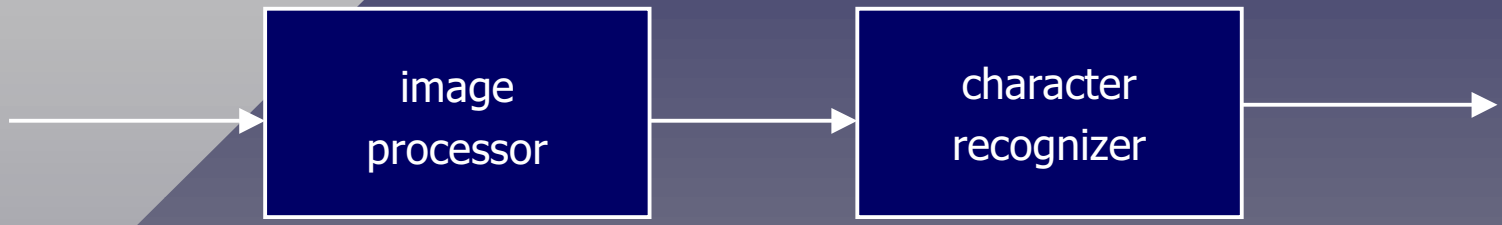
CLPR class diagram



CLPR class diagram



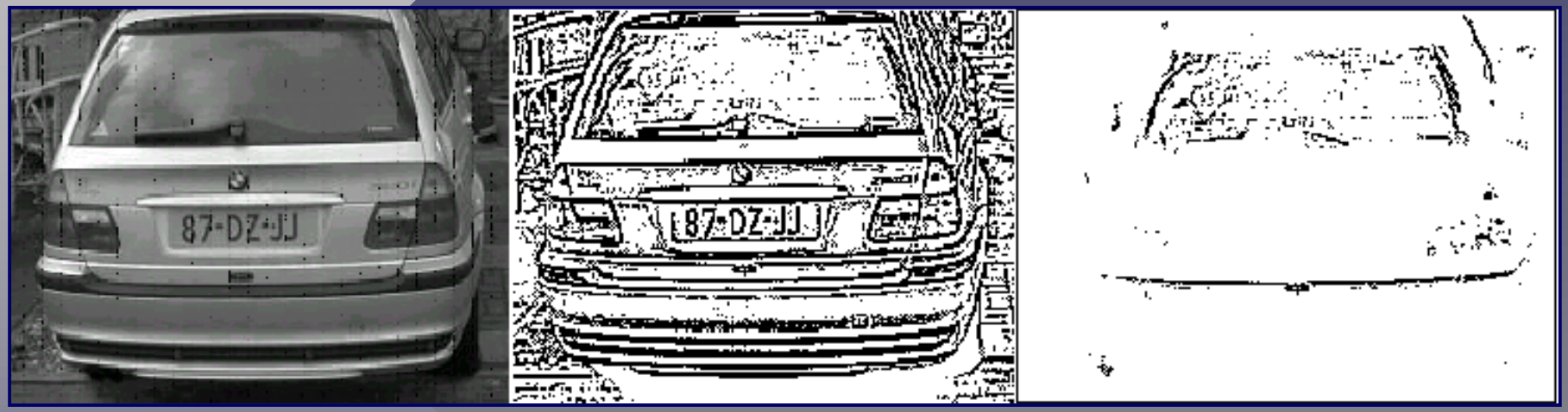
CLPR evaluation.



- Image processor
  - succesfully isolating all plate character : 87 %
- Character recognizer
  - correctly classifying an isolated character : 94 %
  - correctly reading a number plate : 73 %

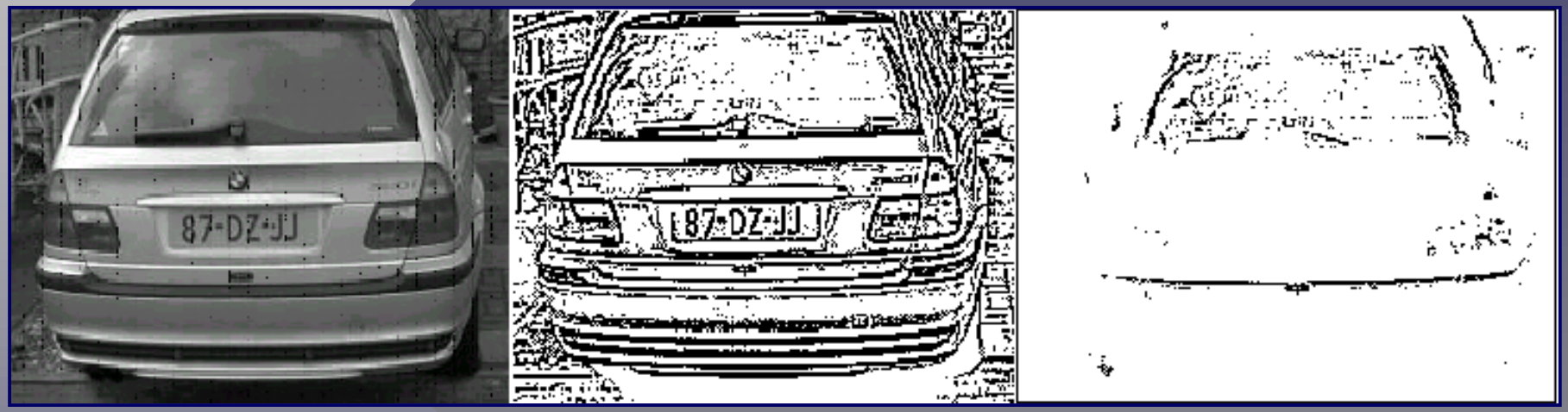
CLPR failures

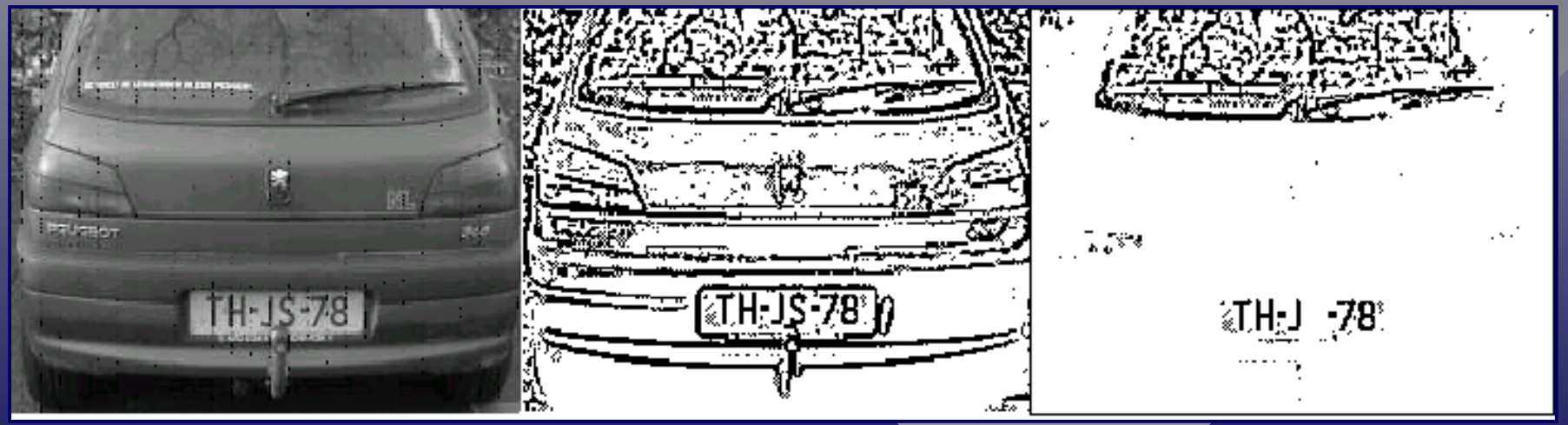
Image processor 13% loss



CLPR failures

Image processor 13% loss

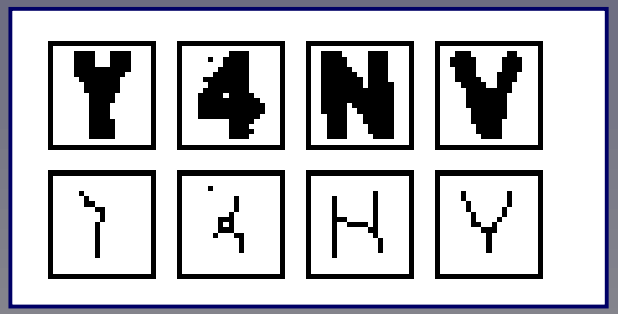


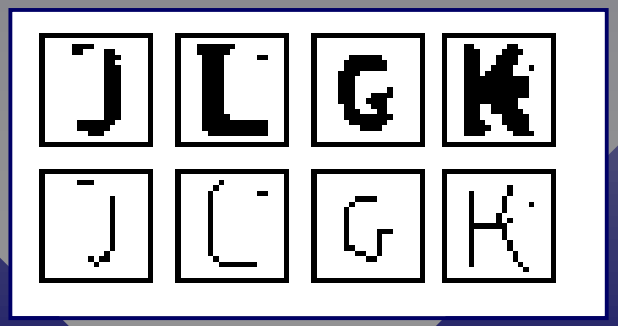


CLPR failures

Recognizer 6% loss

- 0/D, 2/Z,..,8/B misclassifications
- Skeletonizer
- Network





Conclusion

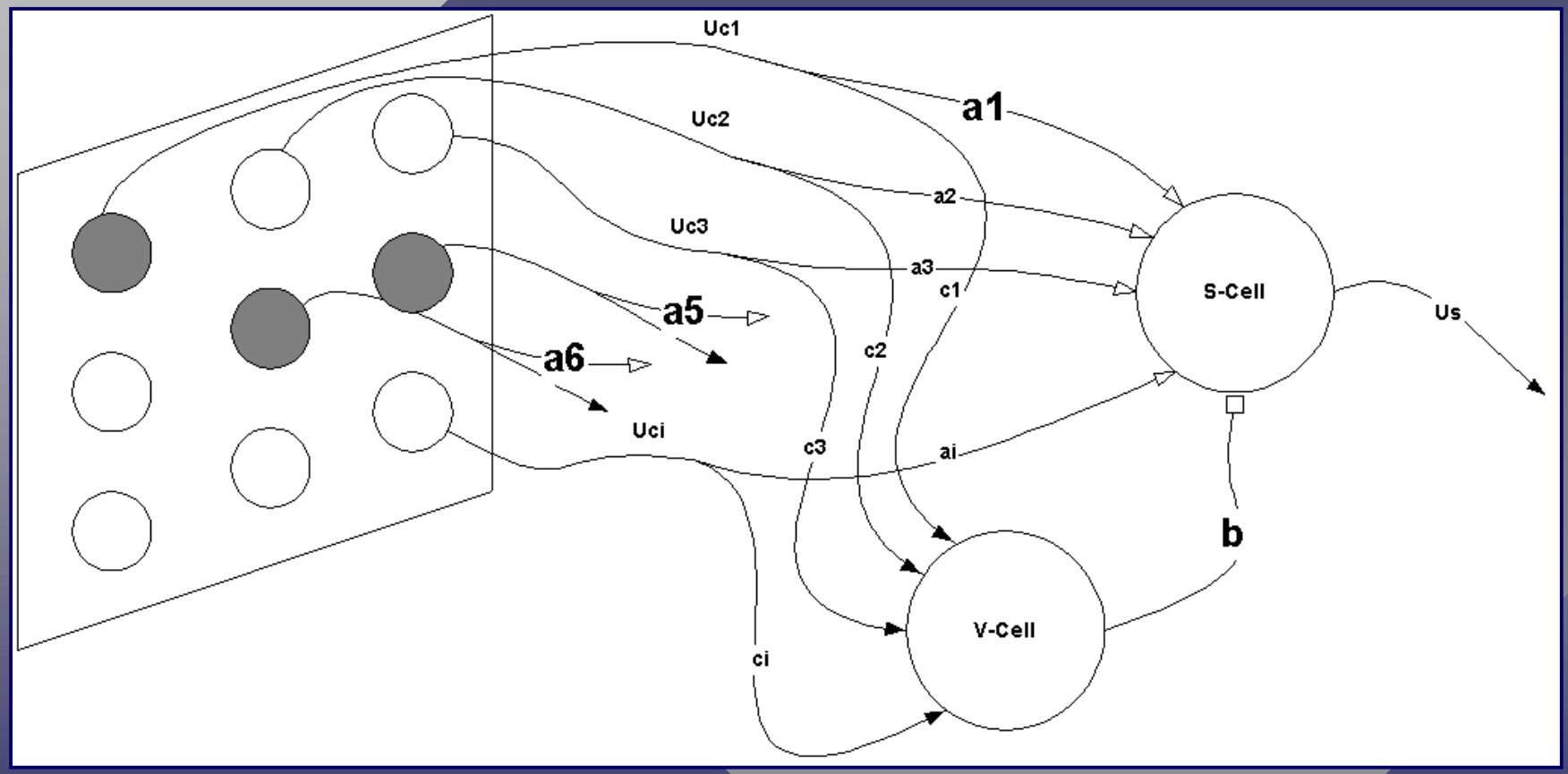
- CLPR system characteristics
  - 65% overall performance
  - down to 10 pixel character sizes. Location, size & color invariant
- Processing Improvements
  - recognizer 73% → 95%
    - post processing using syntax forcing (≈10 %),
    - skeletonizer and neocognitron configuration/training (≈10 %)
  - pre-processor 87% → 95%
- 90% should be achievable using this concept

neocognitron simulator

CLPR prototype

Neocognitron network.

network training.



$$\Delta a_{\lambda}(\nu,\kappa,\rho) = q_{\lambda} c_{\lambda}(\nu) Uc_{\lambda-1(\eta+\nu,\kappa)}$$

$$\Delta b_{\lambda}(\rho) = q_{\lambda} Uv_{\lambda-1(\eta)}$$