MatCnn InstantVision Toolbox for MatLab User's Guide

Eutecus, Inc.

MatCnn InstantVision Toolbox for MatLab User's Guide: 1.0

Eutecus, Inc.

1936 University Avenue Suite #360

Berkeley, CA, US, 94704

Tel. 1-510-540-9603

Fax 1-510-315-2326

http://www.eutecus.com

info@eutecus.com

Table of Contents

1. Introduction

The MatCnn IV Toolbox for MatLab is an easy and flexible test environment for single-layer Cellular Neural/Nonlinear Network (CNN) based simulations.

2. Getting Started

2.1. Installation

Adding MatCnn to the Matlab environment requires to easy steps:

1. unpack the files in an arbitrary directory
2. add this directory with subfolders to the Matlab Path (File menu \ Set Path on the Matlab interface.)

The MatCnn Toolbox has been developed and tested only on MATLAB 4.2 and upper Matlab releases.

It uses the Image Processing Toolbox. Without this toolbox some functions will not work.

2.2. Definitions

The MatCnn IV Toolbox for Matlab builds on top of Matlab using the numerical and visualization environment to provide functions specialized to Cellular Neural Network (CNN) simulations.This subsection gives a short list of the basic CNN terminology:

CNN	Cellular Neural (nonlinear) Network is a regular (rectangular, hexagonal etc.) array of mainly identical dynamical systems, called cells which satisfies two properties: (i) most interactions are local within a finite radius r, and (ii) all state values are continuous valued signals
base cell	The general processing unit of the CNN array : an N-th order dynamical system. In circuit theory and modeling in most cases a first order cell is used consisting of a linear capacitor, a linear resistor, a constant current source and additional voltage controlled current sources (i.e. the interconnections from the neighboring cells). The voltage measured on the linear capacitor is the state value of the base cell. Each cell has its own input and output (the latter is calculated through the output characteristics from the state value, see later).
output characteristics	A general nonlinear (in most cases: sigmoid-type) function defining the output - state connection of a single cell.
CNN layers and images	In its most general form a CNN array is three

	dimensional, but can also be interpreted as a multi-layer two dimensional processing array. If the CNN array is rectangular then all CNN base processing units can be assigned to an image pixel. In this interpretation 5 different “images” can describe a CNN layer, namely the input, state, output, bias, and mask images. The input, state and output image consist of the input, state and output values of all cells in space at a given time. The bias image (also referred to as the bias map) is a space variant description of the constant cell current that can vary from cell to cell in space. The mask image is binary and defines whether a certain cell should operate or not in an analog or logical CNN operation.
template	A template specifies the interaction between each cell and all its neighboring cells in terms of their input, state, and output variables. The inter-cell interaction could be linear, nonlinear or delay type.
analogic algorithm	Analogic is a contraction for analog and logic computation. An analogic algorithm stands for a sequence of analog and logical operations solving a defined (image) processing task.
CNN Universal Machine	The first algorithmically programmable analog array computer having a real time and supercomputing power on a single chip [3]. In a CNN Universal Chip the CNN nucleus contains the analog and logic elements. In addition, five other peripheral units convert this cell nucleus to a universal cell. The base cell is completed with a local analog memory (LAM), a local logical memory (LLM), a local analog output unit (LAOU), a local logical unit (LLU) and a local communication and control unit (LCC). The LLC receives the switch configuration (functionality of the universal cell) from the global switch control register (SCR) and controls the switches in the cell. The LAM and LLM memories facilitate the implementation of several algorithmic steps on signal arrays without global input/output. LAOU and LLM make possible to combine these values. The templates (the analog instructions) are stored in the global analog program register (APR) and local logic functions are available from global logic program register (LPR). Each cell is controlled by the global analogic programming unit (GAPU) consisting of the SCR, APR, LPR and

ACU (analogic control unit).

2.3. Running a CNN simulation

To run a CNN simulation and visualize the result in MATCNN the following steps should be followed:

- set the CNN environment and template library

  e.g:

```
SetEnv;                % set the CNN environment
TemGroup = ‘MyTemLib’;  % the template library is stored in file 'mytemlib.m'
```

  The environment always has to be set. If the template library is not specified the default library of the MatCnn is 'temlib.m'

- initialize the input and state images assigned to the CNN model

  e.g:

```
INPUT1 = LBmp2CNN(‘Road’);     % road.bmp is loaded to the input
STATE = zeros(size(INPUT1));  % all initial state values are set to zero
```

  The input initialization is optional (depends on the template), but the state layer should always be initialized. In this phase noise can also be added to the images for test purposes (see CImNoise).

- specify whether the bias and mask image will be used

  e.g:

```
UseBiasMap = 1;  % use the bias image
UseMask = 0;     % do not use the mask image
```

  If the UseBiasMap global is set to 1 the BIAS image should be initialized, similarly when the UseMask global is set to 1 the MASK image should also be initialized. The SetEnv MatCnn script sets these global variables to zero.

- specify CNN boundary condition

  e.g:

```
Boundary = -1;  % the boundary is set to a constant value (-1)
```

  The boundary condition specified can be constant (-1 <= Boundary <= 1), zero flux (Boundary = 2) and torus (Boundary = 3).

- Set the simulation parameters (time step and number of iteration steps)

  e.g:

```
TimeStep = 0.1;   % the time step of the simulation is 0.1
IterNum = 100;    % the number of iteration steps is 100
```

The default values for TimeStep = 0.2 and IterNum = 25, that correspond approximately to 5t (when R = 1 and C = 1) analog transient length, guarantee that in a non-propagating CNN transient all cells will reach the steady state value. However, for a number of templates different settings should be used.

- load the template that completely determines the CNN model

  e.g:

  ```
  LoadTem('EDGE');   % load the EDGE template from the specified library
  ```

  All templates of a project or algorithm can be stored in a common library (M-file). The LoadTem function activates the given one from the specified library that will determine the CNN model of the simulation.

- run the simulation with the specified template

  e.g:

  ```
  RunTem;    % run the CNN simulation with the specified template
  ```

  The ODE of the CNN model is simulated using the forward Euler formula.

- visualize the result

  e.g:

  ```
  CNNShow(OUTPUT);    % show the output of the CNN simulation
  ```

  Since the CNN is primarily referred to as a visual microprocessor, and the inputs and outputs are images, in most cases it might be helpful to visualize these images using different magnification rates, palettes etc. The user can exploit the capabilities of MATLAB's graphical interface and the Image Processing Toolbox when evaluating the CNN performance.

2.4. Examples

2.4.1. Binary edge detection

A simple example using the EDGE template from the default template library to perform edge detection on a black and white image (road.bmp) stored in user's directory. The output is visualized and saved as roadout.bmp

```
SetEnv;
INPUT1 = LBmp2CNN('Road');
STATE = INPUT1;
Boundary = -1;
TimeStep = 0.1;
IterNum = 50;
LoadTem('EDGE');
RunTem;
CNNShow(OUTPUT);
SCNN2Bmp('RoadOut', OUTPUT);
```

2.4.2. CNN Simulation with a Linear Template

This example performs diffusion:

```
% set CNN environment
 SetEnv;             % default environment
 TemGroup='TemLib';  % default template library
% load images, initialize layers
 load pic2;          % loads the image from pic2.mat to the INPUT1
 STATE = INPUT1;
% set boundary condition
 Boundary = 2;       % zero flux boundary condition
% run linear template
 LoadTem('DIFFUS');  % loads the DIFFUS template (linear)
 TimeStep = 0.2;
 IterNum = 50;
 RunTem;             % runs the CNN simulation
% show result
 subplot(121); CNNShow(INPUT1);    % displays the input
 xlabel('Input');
 subplot(122); CNNShow(OUTPUT);    % displays the output
 xlabel('O: Diffus');
```

Input

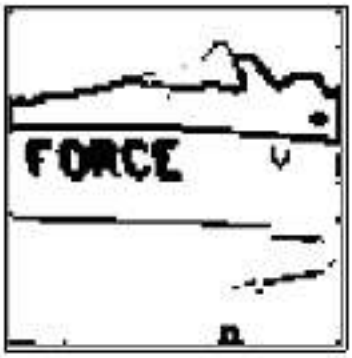
O: Diffus

2.4.3. Sample CNN Simulation with a Nonlinear "AB - type" Template

This example performs gradient detection:

```
% set CNN environment
 SetEnv                      % default environment
 TemGroup = 'TemLib';        % default template library
% load images, initialize layers
 load pic2;                  % loads the image from pic2.mat to the INPUT1
 STATE = INPUT1;
% set boundary condition
 Boundary = 2;               % zero flux boundary condition
% run nonlinear AB template
 LoadTem('GRADT');           % loads the GRADT template (nonlinear "AB-type")
 TimeStep = 0.4;
 IterNum = 15;
 RunTem;                     % runs the CNN simulation
% show result
 subplot(211); cnnshow(INPUT1);    % displays the input
  xlabel('Input');
 subplot(212); cnnshow(OUTPUT);    % displays the output
  xlabel('O: Grad');
```

Input



O: Grad

2.4.4. Sample CNN Simulation with a Nonlinear “D - type” Template

This example performs median filtering:

```
% set CNN environment
 SetEnv                        % default environment
 TemGroup = 'TemLib';          % default template library
% load images, initialize layers
 load pic2;                    % loads the image from pic2.mat to the INPUT1
 STATE = INPUT1;
% put noise in the image
 STATE1 = cimnoise(STATE1, 'salt & pepper',0.05);
 INPUT1 = STATE1;              % 1st input
 INPUT2 = STATE1;              % 2nd input
% set boundary condition
 Boundary = 2;                 % zero flux boundary condition
% run nonlinear D template
 LoadTem('MEDIAN');            % loads the MEDIAN template (nonlinear “D-type”)
 TimeStep = 0.02;
 IterNum = 50;
 RunTem;                       % runs the CNN simulation
% show result
 subplot(211); cnnshow(INPUT1);      % displays the input
  xlabel(‘Input’);
 subplot(212); cnnshow(OUTPUT1);     % displays the output
  xlabel(‘O: Median’);
```

Input

O: Median

3. Quick Reference

3.1. Quick Reference for MatCNN - Analogic CNN Simulation Toolbox for MATLAB

Version: 1.6, last modified on 2005/06/14 03:06:00 by histvan

Basic scripts and functions

setenv	set CNN environment and initialize global variables (script)
showenv	show global variables of the CNN environment (script)
loadtem	load the specified CNN template (function)
showtem	show the actual template loaded into the CNN environment (script)
runtem	run the specified CNN template (function)
temexec	execute template operation (script)
temexecf	execute template operation (function)
logexec	execute logical function (function)
cnnshow	show a CNN-type intensity image (function)
cnnmshow	show multiple CNN-type intensity images (function)
temlib	default CNN template library (script)
transfer	transfer in between two CNN-type image memories (function)

CNN based image processing functions

f_bmorph	binary morphology
f_breconstr	binary object reconstruction
f_dfilt	linear and nonlinear diffusion based filtering
f_edge	edge detection
f_fuzzy	fuzzy decomposition
f_histmod	histgoram modification
f_mean	mean filtering

f_segment	gray-scale image segmentation
f_sfilt	statistic filtering
f_skele	skeletonization
f_sort	sized based binary object sorting
f_thres	thresholding
f_var	variance filtering

Miscellaneous functions

cnn2gray	convert a CNN-type image to a gray-scale intensity image
gray2cnn	convert a gray-scale intensity image to a CNN-type image
cbound	add a specified boundary to a CNN-type image
cimnoise	put noise in a CNN-type image
scnn2bmp	save a CNN-type image to disk in BMP format
lbmp2cnn	load a BMP file from disk and convert it to a CNN-type image
scnn2avi	save a CNN-type image into an avi stream
lavi2cnn	load an avi frame and convert it to a CNN-type image

Basic MEX-files

tlinear	linear CNN template simulation (1st nbr)
tlinear2	linear CNN template simulation (2nd nbr)
tnlinab	nonlinear AB-type CNN template simulation (1st nbr)
tnlind	nonlinear D-type CNN template simulation (2nd nbr)

Special MEX-files implementing nonlinear filters

histmod	histogram modification (CNN t.s. and digital)
tmedian	median (ranked order) CNN template simulation

tmedianh	median (ranked order) CNN t.s. (for switch analysis)
tanisod	single layer models
tanisod2	multi-layer models
ordstat	order statistic (OS) filters (digital)
modfilt	mode filters (digital)

Notes:

- There is no upper level support from the MATCNN environment for this functions.

MATCNN demos

d_lin	linear CNN template demo
d_nlinab	nonlinear AB-type template demo
d_nlind	nonlinear D-type template demo
d_algo	analogic CNN algorithm demo (combining linear and nonlinear templates)
d_algo_ftol	analogic CNN algorithm demo (fault tolerance test)
d_func	demo of all functions in the toolbox
d_matcnn	runs all MATCNN demos
showdpic	show demo pictures of MATCNN (see MAT-files)