Stochastics and Quality Control

Quality Control Using Convolutional Neural Networks Applied to Samples of Very Small Size

Supplement 1

R. A. Chatzimichail, PhD [a] and A. T. Hatjimihail, MD, PhD [a]*

[a] *Hellenic Complex Systems Laboratory, Drama, Greece*

* Corresponding author:
Dr Aristides T. Hatjimihail,
Kostis Palamas 21,
Drama, Greece
66131
ath@hcsl.com

The Convolutional Neural Networks Models

Tables 1 and 2 are summary presentations of their layers and nodes, while Tables 3-6 present the files of 1D - CNN models in various formats.

Table 1. The layers and nodes of the convolutional neural networks

CNN	$N_a(x_1, x_2)$		$N_a(x_1, x_2, x_3)$		$N_a(x_1, x_2, x_3, x_4)$	
	Layers	Nodes	Layers	Nodes	Layers	Nodes
Input	1	2	1	3	1	4
Hidden	67	2102	67	5577	67	11628
Output	1	2	1	2	1	2

Table 2. The input and output of the convolutional neural networks

CNN	$N_a(x_1, x_2)$	$N_a(x_1, x_2, x_3)$	$N_a(x_1, x_2, x_3, x_4)$
Input	(x_1, x_2)	(x_1, x_2, x_3)	(x_1, x_2, x_3, x_4)
Output: Decoder	$(False, True)$	$(False, True)$	$(False, True)$

$False \rightarrow$ conforming to quality specifications, $True \rightarrow$ nonconforming to quality specifications

Table 3. The files of the graphs of the convolutional neural networks' models in SVG and PDF formats

CNN	SVG	PDF
$N_a(x_1, x_2)$	/CNN/SVG /nn_2. svg	/CNN/PDF/nn_2.pdf
$N_a(x_1, x_2, x_3)$	/CNN/SVG /nn_3. svg	/CNN/PDF/nn_3.pdf
$N_a(x_1, x_2, x_3, x_4)$	/CNN/SVG /nn_4. svg	/CNN/PDF/nn_4.pdf

The graphs are based on ONNX framework models. As ONNX does not support decoders, please see Table 2

Table 4. The files of the exported trained convolutional neural networks' models for the Apache MXNet framework

CNN	MXNet	
$N_1(x_1, x_2)$	/CNN/MXNet/nn1_2.json	/CNN/MXNet/nn1_2.params
$N_1(x_1, x_2, x_3)$	/CNN/MXNet/nn1_3.json	/CNN/MXNet/nn1_3.params
$N_1(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn1_4.json	/CNN/MXNet/nn1_4.params
$N_2(x_1, x_2)$	/CNN/MXNet/nn2_2.json	/CNN/MXNet/nn2_2.params
$N_2(x_1, x_2, x_3)$	/CNN/MXNet/nn2_3.json	/CNN/MXNet/nn2_3.params
$N_2(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn2_4.json	/CNN/MXNet/nn2_4.params
$N_3(x_1, x_2)$	/CNN/MXNet/nn3_2.json	/CNN/MXNet/nn3_2.params
$N_3(x_1, x_2, x_3)$	/CNN/MXNet/nn3_3.json	/CNN/MXNet/nn3_3.params
$N_3(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn3_4.json	/CNN/MXNet/nn3_4.params
$N_4(x_1, x_2)$	/CNN/MXNet/nn4_2.json	/CNN/MXNet/nn4_2.params
$N_4(x_1, x_2, x_3)$	/CNN/MXNet/nn4_3.json	/CNN/MXNet/nn4_3.params
$N_4(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn4_4.json	/CNN/MXNet/nn4_4.params
$N_6(x_1, x_2)$	/CNN/MXNet/nn6_2.json	/CNN/MXNet/nn6_2.params
$N_6(x_1, x_2, x_3)$	/CNN/MXNet/nn6_3.json	/CNN/MXNet/nn6_3.params
$N_6(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn6_4.json	/CNN/MXNet/nn6_4.params
$N_8(x_1, x_2)$	/CNN/MXNet/nn8_2.json	/CNN/MXNet/nn8_2.params
$N_8(x_1, x_2, x_3)$	/CNN/MXNet/nn8_3.json	/CNN/MXNet/nn8_3.params
$N_8(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn8_4.json	/CNN/MXNet/nn8_4.params
$N_{12}(x_1, x_2)$	/CNN/MXNet/nn12_2.json	/CNN/MXNet/nn12_2.params
$N_{12}(x_1, x_2, x_3)$	/CNN/MXNet/nn12_3.json	/CNN/MXNet/nn12_3.params
$N_{12}(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn12_4.json	/CNN/MXNet/nn12_4.params
$N_{16}(x_1, x_2)$	/CNN/MXNet/nn16_2.json	/CNN/MXNet/nn16_2.params
$N_{16}(x_1, x_2, x_3)$	/CNN/MXNet/nn16_3.json	/CNN/MXNet/nn16_3.params
$N_{16}(x_1, x_2, x_3, x_4)$	/CNN/MXNet/nn16_4.json	/CNN/MXNet/nn16_4.params

As MXNet model definitions do not preserve information about input and output dimensions, please see Table 2

Table 5. The files of the exported trained convolutional neural networks' models for the Apache MXNet, and the Wolfram Language WMLF frameworks

CNN	ONNX	WMLF
$N_1(x_1, x_2)$	/CNN/ONNX/nn1_2.onnx	/CNN/WMLF/nn1_2.wmlf
$N_1(x_1, x_2, x_3)$	/CNN/ONNX/nn1_3.onnx	/CNN/WMLF/nn1_3.wmlf
$N_1(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn1_4.onnx	/CNN/WMLF/nn1_4.wmlf
$N_2(x_1, x_2)$	/CNN/ONNX/nn2_2.onnx	/CNN/WMLF/nn2_2.wmlf
$N_2(x_1, x_2, x_3)$	/CNN/ONNX/nn2_3.onnx	/CNN/WMLF/nn2_3.wmlf
$N_2(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn2_4.onnx	/CNN/WMLF/nn2_4.wmlf
$N_3(x_1, x_2)$	/CNN/ONNX/nn3_2.onnx	/CNN/WMLF/nn3_2.wmlf
$N_3(x_1, x_2, x_3)$	/CNN/ONNX/nn3_3.onnx	/CNN/WMLF/nn3_3.wmlf
$N_3(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn3_4.onnx	/CNN/WMLF/nn3_4.wmlf
$N_4(x_1, x_2)$	/CNN/ONNX/nn4_2.onnx	/CNN/WMLF/nn4_2.wmlf
$N_4(x_1, x_2, x_3)$	/CNN/ONNX/nn4_3.onnx	/CNN/WMLF/nn4_3.wmlf
$N_4(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn4_4.onnx	/CNN/WMLF/nn4_4.wmlf
$N_6(x_1, x_2)$	/CNN/ONNX/nn6_2.onnx	/CNN/WMLF/nn6_2.wmlf
$N_6(x_1, x_2, x_3)$	/CNN/ONNX/nn6_3.onnx	/CNN/WMLF/nn6_3.wmlf
$N_6(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn6_4.onnx	/CNN/WMLF/nn6_4.wmlf
$N_8(x_1, x_2)$	/CNN/ONNX/nn8_2.onnx	/CNN/WMLF/nn8_2.wmlf
$N_8(x_1, x_2, x_3)$	/CNN/ONNX/nn8_3.onnx	/CNN/WMLF/nn8_3.wmlf
$N_8(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn8_4.onnx	/CNN/WMLF/nn8_4.wmlf
$N_{12}(x_1, x_2)$	/CNN/ONNX/nn12_2.onnx	/CNN/WMLF/nn12_2.wmlf
$N_{12}(x_1, x_2, x_3)$	/CNN/ONNX/nn12_3.onnx	/CNN/WMLF/nn12_3.wmlf
$N_{12}(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn12_4.onnx	/CNN/WMLF/nn12_4.wmlf
$N_{16}(x_1, x_2)$	/CNN/ONNX/nn16_2.onnx	/CNN/WMLF/nn16_2.wmlf
$N_{16}(x_1, x_2, x_3)$	/CNN/ONNX/nn16_3.onnx	/CNN/WMLF/nn16_3.wmlf
$N_{16}(x_1, x_2, x_3, x_4)$	/CNN/ONNX/nn16_4.onnx	/CNN/WMLF/nn16_4.wmlf

As ONNX does not support decoders, please see Table 2

Table 6. The files of the trained convolutional neural networks as Wolfram Language objects

CNN	Wolfram Language Objects		
	Name	Files	
$N_1(x_1, x_2)$	"ncl2"	/CNN/WLO/NN1/ncl2/object.wl	/CNN/WLO/NN1/ncl2/put.wl
$N_1(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN1/ncl3/object.wl	/CNN/WLO/NN1/ncl3/put.wl
$N_1(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN1/ncl4/object.wl	/CNN/WLO/NN1/ncl4/put.wl
$N_2(x_1, x_2)$	"ncl2"	/CNN/WLO/NN2/ncl2/object.wl	/CNN/WLO/NN2/ncl2/put.wl
$N_2(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN2/ncl3/object.wl	/CNN/WLO/NN2/ncl3/put.wl
$N_2(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN2/ncl4/object.wl	/CNN/WLO/NN2/ncl4/put.wl
$N_3(x_1, x_2)$	"ncl2"	/CNN/WLO/NN3/ncl2/object.wl	/CNN/WLO/NN3/ncl2/put.wl
$N_3(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN3/ncl3/object.wl	/CNN/WLO/NN3/ncl3/put.wl
$N_3(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN3/ncl4/object.wl	/CNN/WLO/NN3/ncl4/put.wl
$N_4(x_1, x_2)$	"ncl2"	/CNN/WLO/NN4/ncl2/object.wl	/CNN/WLO/NN4/ncl2/put.wl
$N_4(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN4/ncl3/object.wl	/CNN/WLO/NN4/ncl3/put.wl
$N_4(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN4/ncl4/object.wl	/CNN/WLO/NN4/ncl4/put.wl
$N_6(x_1, x_2)$	"ncl2"	/CNN/WLO/NN6/ncl2/object.wl	/CNN/WLO/NN6/ncl2/put.wl
$N_6(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN6/ncl3/object.wl	/CNN/WLO/NN6/ncl3/put.wl
$N_6(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN6/ncl4/object.wl	/CNN/WLO/NN6/ncl4/put.wl
$N_8(x_1, x_2)$	"ncl2"	/CNN/WLO/NN8/ncl2/object.wl	/CNN/WLO/NN8/ncl2/put.wl
$N_8(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN8/ncl3/object.wl	/CNN/WLO/NN8/ncl3/put.wl
$N_8(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN8/ncl4/object.wl	/CNN/WLO/NN8/ncl4/put.wl
$N_{12}(x_1, x_2)$	"ncl2"	/CNN/WLO/NN12/ncl2/object.wl	/CNN/WLO/NN12/ncl2/put.wl
$N_{12}(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN12/ncl3/object.wl	/CNN/WLO/NN12/ncl3/put.wl
$N_{12}(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN12/ncl4/object.wl	/CNN/WLO/NN12/ncl4/put.wl
$N_{16}(x_1, x_2)$	"ncl2"	/CNN/WLO/NN16/ncl2/object.wl	/CNN/WLO/NN16/ncl2/put.wl
$N_{16}(x_1, x_2, x_3)$	"ncl3"	/CNN/WLO/NN16/ncl3/object.wl	/CNN/WLO/NN16/ncl3/put.wl
$N_{16}(x_1, x_2, x_3, x_4)$	"ncl4"	/CNN/WLO/NN16/ncl4/object.wl	/CNN/WLO/NN16/ncl4/put.wl