ISSN 2007-9737

Development of a Platform for Generation of CNN and Multilayer Neural Networks

Daniel Marcelo González-Arriaga[1], María Aurora Diozcora Vargas-Treviño[2],
Josefina Guerrero García [1], Jesús López Gómez[3]

[1] Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Computación,
México

[2] Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Electrónica,
México

[3] Universidad Juárez Autónoma de Tabasco,
División Académica de Ingeniería y Arquitectura,
México

daniel.gonzaleza@alumno.buap.mx,
{aurora.vargas,josefina.guerrero}@correo.buap.mx, jesus.lopezg@ujat.mx

Abstract. This research presents the design of a platform that assists in the generation of convolutional (CNN) and multilayer neural networks to provide a user-friendly interface for the design, formation, and development of neural networks. This platform is developed in LabVIEW as this software allows to perform inter-faces and generate an executable for use. It aims to reduce the development time of neural networks by providing an assistant-like graphical interface that guides the user through various common scenarios (data import, neural network construction and adjustment), allows the user to focus on solving their problems without having to write code, edit text files, or manually analyze recorded data. The user interface with the options offered is described. The way the neural network is generated is described. The results generated with the platform are presented producing an image with the proposed methodology applying a complete convolution layer. The usefulness of this platform is explained by presenting a case where there is a significant improvement in the development of a neural network, in time and reduction of errors.

Keywords: CNN, multilayer, software, platform, LabVIEW platform, MATLAB platform.

1 Introduction

The artificial neural network is a model extracted from the biological neural network. In the biological neural network, "when a neuron receives an exciting input that is large enough compared to its inhibitory input, it sends a peak of electrical activity through its axon. Learning occurs by changing the effectiveness of synapses so that the influence of one neuron on another changes [1].

Artificial intelligence is becoming a widely used tool for its robust applicability to problems, particularly those that cannot be solved well by humans, for example, in medicine where algorithms are used to identify subjects with a family history of an inherited disease or an increased risk of a chronic disease or in the evaluation of changes in human performance in such situations-rehabilitation [2].

There is a particular type of artificial neural network that makes a difference in practice, which is precisely the one that corresponds to networks used to process signals: convolutional networks. Their success in solving computer vision problems

doi: 10.13053/CyS-26-1-4162

that, until recently, were considered almost intractable, has served to reaffirm neural networks in Artificial Intelligence. Recent advances in CNNs have led to vast improvements in the accuracy of hearing and vision systems. Characterized by deep structures and many parameters, deep CNNs challenge the performance of current computers. Most of the work is focused on the implementation and improvement of these networks.

In recent years, many advances have been made with deep neural networks, delivering cutting-edge results in machine learning tasks such as the classification of images and documents. Companies make use of the latest advances in deep learning by benefiting from a rapidly evolving software environment comprising different deep learning frameworks.

Tensor Flow [3] began in 2011 as an internal Google project called "Google Brain" and was made public in 2017 as an open-source deep learning system, meaning a neural network, which can run on multiple CPUs and GPUs. It is used to train neural networks that can detect and decipher patterns and correlations analogous to those we see in human learning and reasoning.

Caffe [4] gives scientists and practitioners a clean, customizable structure for cutting-edge deep learning algorithms and a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB links to train and implement general-purpose, convolutional neural networks and other deep models efficiently in commodity architectures.

ONNX [5] (Open Neural Network Exchange) was announced just in September 2017. It's a joint effort by Microsoft and Facebook. ONNX is a format designed to make it easy to exchange deep learning models between people of this nature. The initiative aims to make it easier for developers to use multiple in-turn neural network programming.

Weka [6] is a software platform for machine learning and data mining written in Java and developed at Waikato University. It is mainly used to make classifiers.

PyTorch [7] is an open-source machine learning library that specializes in tensor calculations, automatic differentiation, and GPU acceleration. For those reasons, PyTorch is one of the most popular deep learning libraries, competing with Keras and TensorFlow for the "most used" deep learning package award. PyTorch tends to be especially popular among the research community because of its Pythonic nature and extensibility facility (i.e., enhancement of custom layer types, network architectures, etc.).

Microsoft Cognitive Toolkit [8] (CNTK) is an open-source toolkit for commercially distributed deep learning. It describes neural networks as a series of computational steps through a directed graph. CNTK allows the user to easily perform and combine popular model types such as breakthrough DNN, convolutional neural networks (CNN) and recurrent neural networks (RNN / LSTM). CNTK implements stochastic gradient downgradient (SGD, error backpropagation) learning with automatic differentiation and parallelization on multiple GPUs and servers.

Corporations like Amazon, Apple, Google, IBM and Microsoft, provide their machine learning services in the cloud, both in the form of pre-trained that can be used for predictions and form platforms that design models based on customer-provided data.

Datalore represents Jetbrains vision of online machine learning environments, which consists of intelligent encoding support for Python online Jupyter laptops running powerful CPU and GPU code, offering real-time collaboration and results sharing facilities [9].

Maintaining a rigorous approach to the life cycle and evolution of machine learning can be difficult, especially when it comes to designing a new machine learning mechanism or process. Access to appropriate testing tools and resources is essential for researchers operating in this field. As a neural network developer, we probably all spend hours customizing a work environment at least once. We recognized that the tools, tools and techniques are more or less the same. Therefore, maintaining a custom environment while keeping all software up to date can be a monotonous task.

However, a local platform is needed for researchers and scientists who wish to experiment and develop this technology, which, unlike those mentioned above, is friendly to people from other areas of research and which must incorporate the advantages of neural networks in their lines of research.

This work aims to help in the reduction in the development time of a neural network, grant ease

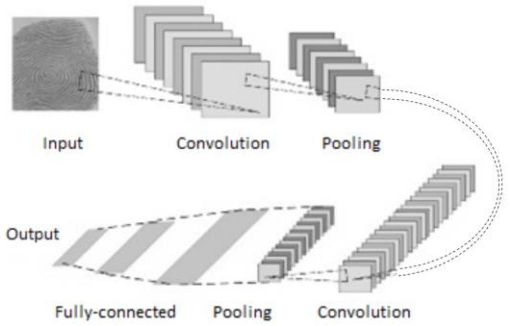


Fig. 1. Structure of a CNN

in modifying the network to find the appropriate configuration for each application. The same purpose of a neural network for example the parametric robot identification that can be performed with neural networks changes between each type of robot, the developed software allows to readjust the design without the need to modify a code of hundreds of lines, by simply modifying the parameters that the user enters in the software, the desired result can be obtained.

This article is divided into 6 chapters. Chapter 2 details the approach of the proposed silver-form and its comparison with other software, chapter 3 details the software, the design of the user interface and the way in which the platform develops the neural network based on the parameters entered by the user, Chapter 4 presents results obtained with the platform, chapter 5 presents an application that could be improved with the use of our software and chapter 6 presents the conclusions.

2 Background

The proposed wizard is designed to be used by someone with basic knowledge of neural networks, for the user to understand the terminology of the parameters needed to configure each layer of the new network and understand each part that composes it. Software like Knime [10] which is developed on the Eclipse platform and programmed, essentially, in Java.

It is conceived as a graphical tool and has a series of nodes (encapsulating different types of algorithms) and arrows (representing data flows) that are displayed and combined in a graphical and interactive manner.

Barista [11], an open-source graphical tool for deep neural network design. Barista uses Caffe as the underlying because of its concept of network layers as the building blocks of a model.

Expresso [12] a GUI tool written in Python is built on top of Caffe. Expresso provides a convenient wizard as a graphical interface that guides the user through various common scenarios, data import, construction, and deep networking, performing various experiments, analyzing, and visualizing the results of these experiments.

These platforms expose deep learning in a GUI and use the Python TensorFlow, Keras5 or Caffe libraries as backends. The user must configure Python manually using virtual environments or Anaconda6 and may have to resolve dependencies of the corresponding deep learning framework. On the contrary, our software is easily installed via the Windows installer.

For the development the platform was analyzed an existing CNN to have the bases of the elements that should be able to integrate the proposed platform.

Figure 1 shows the general structure of a CNN, to the image input an image processing is performed that can contain, filtering, evaluation function, bias and so on, then there is the pooling layer that reduces the size of the image, after repeating these layers several times the information enters a fully connected neural network and finally the output is obtained.

In the construction of a convolutional neural network, the more layers are added, the more complex the combinatorics of outputs and inputs in the process become. For example, a neural network with 3 convolutional layers, the first layer is shown in Figure 2, 10 9x9 kernels are applied and a sub-sample is made by obtaining a map of 46x46x10 on the output.

Finally, the output of convolutional layer 3 is connected to a fully connected network that has 3610 input neurons that correspond to each value of the output map. Multiple hidden layers are built until only 4 parameters are output.

Mathematically, convolution is a mathematical operation performed on two functions to produce a

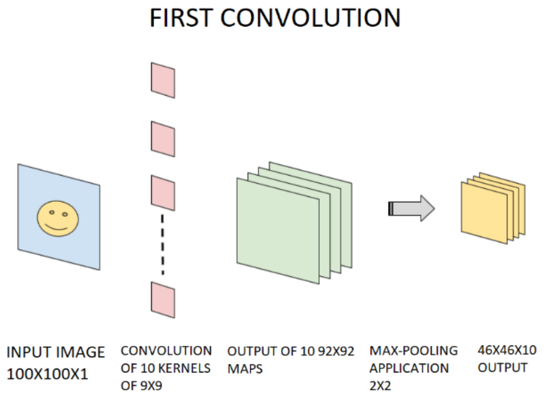


Fig. 2. First layer of convolution

Table 1. Data from subsequent convolutions

Layer	Input	Kernel size	Pooling	Output
2	46X46X10	5X5X10	2X2	21X21X10
3	21X21X10	3X3X10	2X2	19x19x10

third that is usually interpreted as a modified (filtered) version of one of the original functions.

Subsequent convolutions are shown in Table 1. Figure 3 illustrates the convolution operation, takes a window of the input image X of the same size of the filter W, and multiplies by its matrix.

The values obtained from the multiplication of each element of the image for each element of the filter are added and the element of the matrix Y containing the resulting elements is obtained, this process is repeated until the entire input map is completed.

During the convolution process you have a small reduction that is related to the size of the filter, this happens when you start to get the window of the input image, being an array of for example 9x9, if you take it from the first reading as center of the matrix, there are no elements in the first half of the matrix, so there is a reduction in the size of the filter minus one, in the given example, if you have a 100x100 input and apply a 9x9 filter you will have an output of 92x92.

Sharing parameters while using local receptive fields allows you to control the number of parameters of a convolutional network. However, the computational cost of processing images in a neural network can be quite high.

For this reason, convolutional networks often include a second key component: pooling or sub-sampling layers.

When a sub-sample is made, the jump size defines how the window is traversed in the original matrix, figure 4 shows a matrix of 4 x 4 and as the window with the jump size 1 is selected, figure 5 shows the result obtained

In comparison, figure 6 where the window break is 2, the number of steps you must perform to complete the path of the same matrix is greater, this modifies the size of the output matrix shown in figure 7.

The window size modifies the size of the sub matrix that is selected to get the maximum average depending on the type of reduction used.

This modifies the result obtained as more elements are taken from the array, but it is still reduced to a single element per selected window, figure 8 shows a window of size 2 with jump 1, and figure 9 shows a window of size 3 with jump 1.

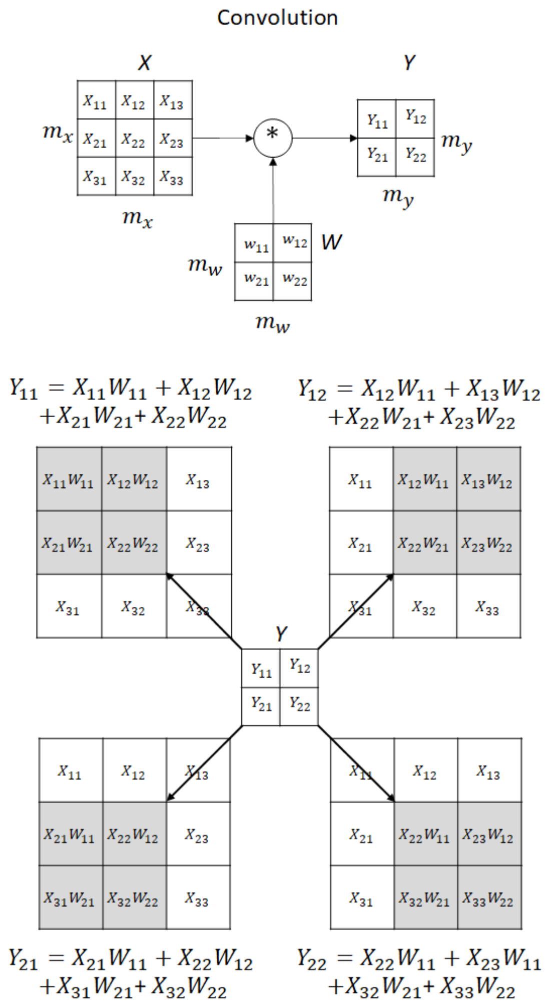
The window and jump size influence the output size of the subsample, it is important to note this for the next layer.

3 Software Framework

The wizard is built in LabVIEW and with-poured into an executable file; the file that is obtained as output has the extension. “.m” can therefore be run on MATLAB or GNU Octave open-source software. The supported neural networks are CNN and multilayer.

Using GNU Octave to run the neural network designed with the platform has the advantage of being a free-to-use program. To use it on Windows, simply download the installer from your website and run it. In the case of MacOS you download the file with extension ".dmg" from the website [13], this will install it as an application, finally, you need to enter the privacy and security settings and unlock the program, this happens because the system detects that it is an application from an unknown developer.

The installation on Linux depends on the distribution, in which you want to install, there is support for these distributions [14]: Debian, Ubuntu, Arch Linux, Slackware, Gentoo, Fedora, openSUSE, CentOS y RHEL.



$$Y_{11} = X_{11}W_{11} + X_{12}W_{12} + X_{21}W_{21} + X_{22}W_{22}$$

$$Y_{12} = X_{12}W_{11} + X_{13}W_{12} + X_{22}W_{21} + X_{23}W_{22}$$

$$Y_{21} = X_{21}W_{11} + X_{22}W_{12} + X_{31}W_{21} + X_{32}W_{22}$$

$$Y_{22} = X_{22}W_{11} + X_{23}W_{11} + X_{32}W_{21} + X_{33}W_{22}$$

Fig. 3. Convolution process

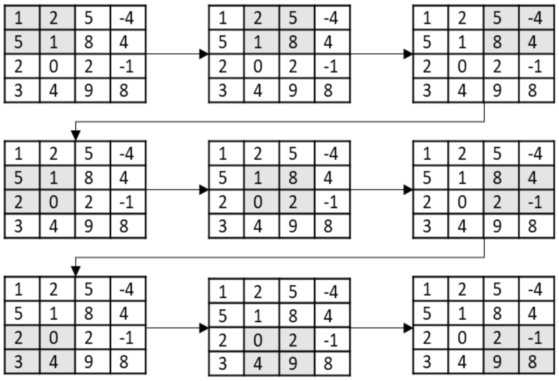


Fig. 4. Window selection with jump size 1

5	8	8
5	8	8
4	9	9

Fig. 5. Sub-sample result with jump 1

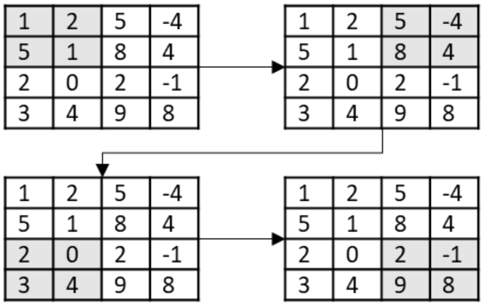


Fig. 6. Window selection with jump size 2

5	8
4	9

Fig. 7. Sub sample result with jump 2

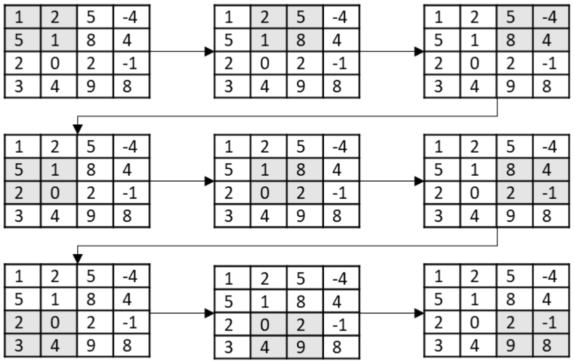


Fig. 8. Sub sample with window size 2

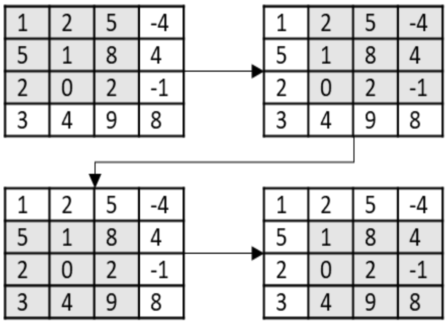
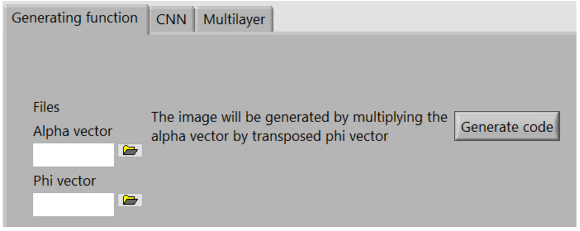


Fig. 9. Sub sample with window size 3

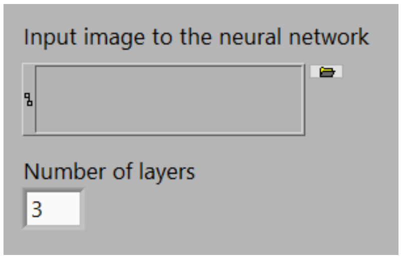


Alpha vector

Phi vector

The image will be generated by multiplying the alpha vector by transposed phi vector

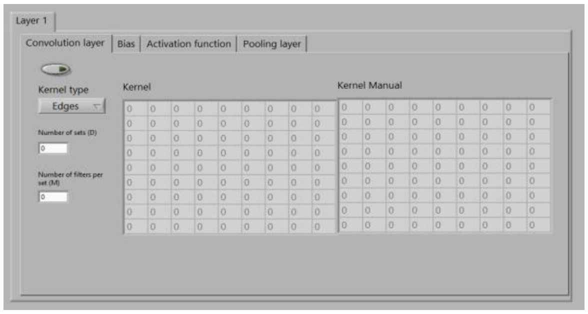
Fig. 10. Generating function user interface



Input image to the neural network

Number of layers
3

Fig. 11. Layer number selection interface and input image selection



Kernel type
Edges

Number of sets (D)
0

Number of filters per set (M)
0

Kernel

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Kernel Manual

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Fig. 12. User interface to create a convolutional network

Installation in Debian and Ubuntu, which are the most used distributions are:

1. Open terminal,
2. Add repository ‘*sudo apt-get install octave*’,
3. Enter the system password,
4. You can additionally install:
    a. *octave-doc*, *octave-info*, and *octave-htmldoc* for the documentation.
    b. *liboctave-dev* for the octave development header files and mkoctfile (required to install Octave Forge packages).
    c. *octave-dbg* for the debugging symbols.

3.1 Graphical User Interface

Figure 10 shows the user interface that as a first option allows the user to select between the options: generating function, convolutional network, and multilayer network. For this interface design tabs were used for the user to select the desired option.

In the first option, the user is asked to select the files containing the vectors ϕ and α to generate an image that can serve as input for CNN.

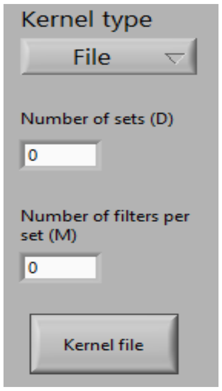
The phi and alpha vectors must be the same size and belong to the real numbers. The operation performed with these vectors is a cross product of Alpha per phi transposed, this generates an array that normalizes between 0 and 255 to be able to be converted to an image of the same size of the array where each element of the array is a pixel of the generated image.

The second option allows to generate the code for a convolutional network, as seen in figure 11 the interface requests the path where the input image for the network is located.

The user is allowed to select the number of convolutional layers the neural network is intended to contain, as the user changes the number of layers desired the interface displays the tabs of each layer so that the user configures each and requests to select the input image as shown in figure 12.

Within each tab of each layer, there are four tabs where the user configures the parameters of each convolutional layer (filtering, bias, activation function and sub-sampling layer) and whether activate each layer type, where the user can choose the layer type.

The first option is convolutional layer where you can select the number of filters you want to apply, how many sets of filters and how many filters for each set, the kernel with which you want the convolution to be done, can be selected from pre-designed kernels, the desired kernel can be manually entered, random kernel can be selected from a defined range or imported from an external file as shown in figure 13. Figure 13 and 14 shows how the interface changes if a random kernel or kernel is selected from a file.

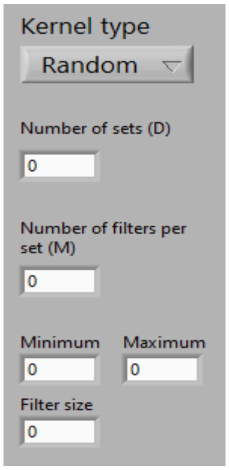


Kernel type: File

Number of sets (D): 0

Number of filters per set (M): 0

Fig. 13. Kernel interface from file



Kernel type: Random

Number of sets (D): 0

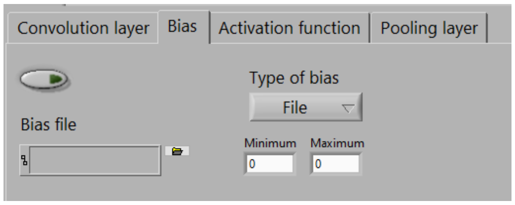
Number of filters per set (M): 0

Minimum: 0

Maximum: 0

Filter size: 0

Fig. 14. Random kernel interface

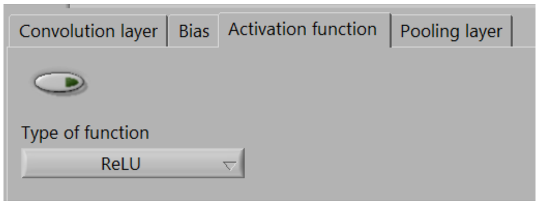


Type of bias: File

Minimum: 0

Maximum: 0

Fig. 15. User interface for bias selection



Type of function: ReLU

Fig. 16. User interface for activation function selection

When random kernel is selected, the filter size and the minimum and maximum range of values are asked. In the case of the kernel from file is requested to select the local file containing the kernel, in case more than one filter is being used, the files must have the same name and numbered at the end of the name, each file will correspond to a kernel for example kernel_1, kernel_2 and so on.

Figure 15 shows the interface in bias settings, in this layer you can select a random selection range or import an external file.

Figure 16 shows the selection of activation function that can be performed, can be selected between the ReLU function, sigmoid function, or hyperbolic tangent.

Figure 17 shows the configuration of the sub-sampling stage that the user can perform, you can select the type of sub-sampling, average or maximum, you can set the window size and the jump size.

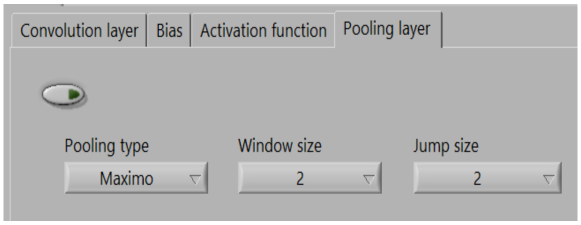
1.1. Generation of the Neural Network

The program generates a MATLAB file with extension. m in the same folder of the system where the input image is located and with the same name for easy identification in the system.

The generated neural network can be executed by CPU or GPU, this gives us greater versatility, both for users who want to optimize the execution times and use the GPU, as for those who do not have a special hardware and use the CPU.

The wizard allows you to create a given image that is not counted with one as input for CNN, this image is created with the vectors ϕ and α that the user provides to generate an image that can serve as input for the CNN, having the address of the vector data read as MATLAB variables and are created as vectors, vector multiplication is performed, and an image is created with the map of values obtained where each resulting value equals a pixel.

The wizard creates the code that allows MATLAB to have the image information, takes the image address provided by the user and using the function "imread" the image information is imported, then converts the image values to double precision to work with them and obtains the image size and saves a variable with the original information in case you need to compare with the original



Pooling type: Maximo
Window size: 2
Jump size: 2

Fig. 17. User interface for sub-sampling configuration

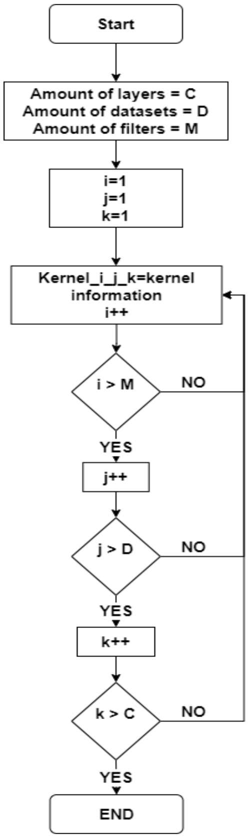


Fig. 18. Writing flow diagram of kernels

information, this process can be repeated for n images.

The handling of the different kernels is of utmost importance as you can get to use a lot. We propose that at the start of the generated network all kernels to be used are declared using the following nomenclature:

$$\text{Kernel\_C\_D\_M} .$$

Here C is the layer number, D is the set number and M is the filter number within the set, this way when the convolution is done only the correct sub-indices are used.

Figure 18 shows the process performed by the platform to write kernels to be used during the neural network. First the desired kernel type is identified (random, by file or predefined), if random values are generated and indexes are placed to identify it, if by file is read with the address provided by the user, the information is imported and the kernel is generated with the corresponding subscripts, if it is pre-defined from the list provided by the software, the values are written.

Using the kernels written, the platform performs the convolution operation described in section 3.

Having the output of the convolutional layer, the sum of Bias is performed,

The value of Bias controls how predisposed the neuron is to shoot a 1 or a 0 independent of the weights. A high bias makes the neuron require a higher input to generate an output of 1. A low bias makes it easier. Figure 19 shows the procedure, adds the input element with a bias value and generates the output matrix.

The next step is the evaluation using a non-linear function, in this process the evaluation of each element of the matrix is performed and a new matrix is obtained with the results of the evaluation, this matrix maintains the same dimension, there are different evaluation functions that are used repeatedly, Figure 20 shows some functions and their evaluation process.

The last procedure performed on the layer is the reduction or pooling, this process was explained in section 3.

4 Results and Discussion

Two files were generated with 100 number alloys each, were entered as ϕ and α vectors in the software to use the image generating function, in figure 21 the generated image.

In Figure 23 we can see the MATLAB output when running the code generated with the platform, as we can see for this case, we selected the edge preset kernel that generates an image with the edges in white and the rest of the image in black, In Figure 24 the convolution with the focus kernel was performed. In both cases only the convolution with the selected kernel applies.

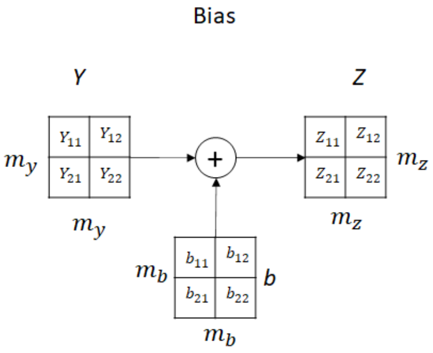


Fig. 19. Bias process

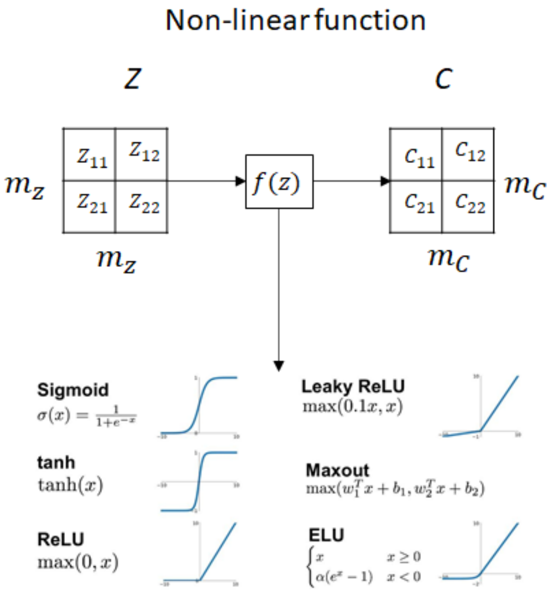


Fig. 20. Assessment of non-linear function

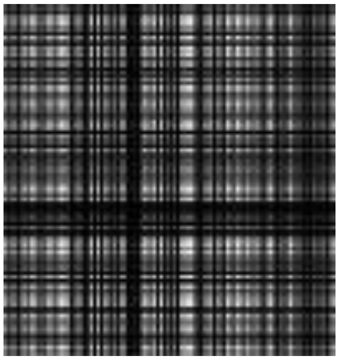


Fig. 21. Generated figure

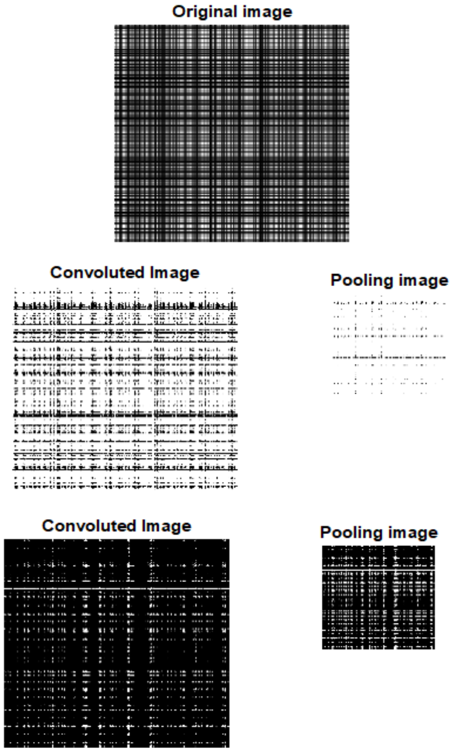


Fig. 22. Convolutional layer applied to generated image

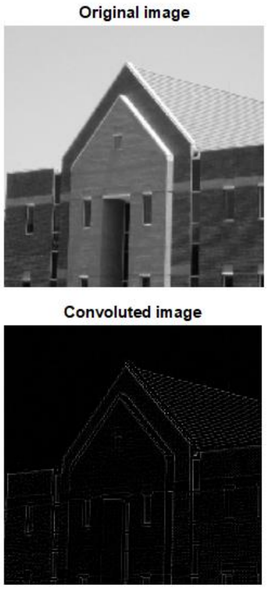


Fig. 23. Image convoluted with focus kernel

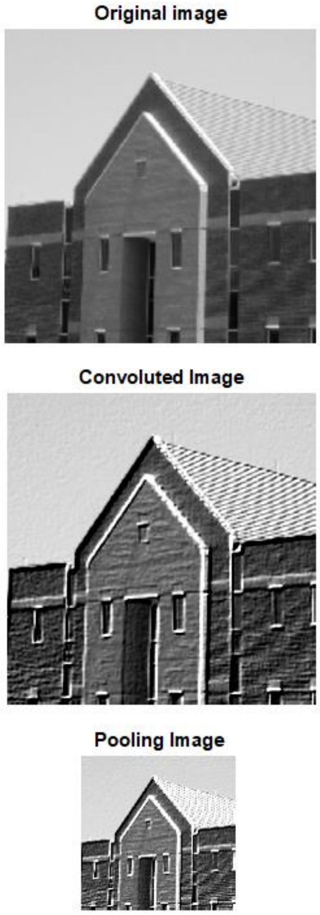


Fig. 24. Image result after a convolutional layer with all components

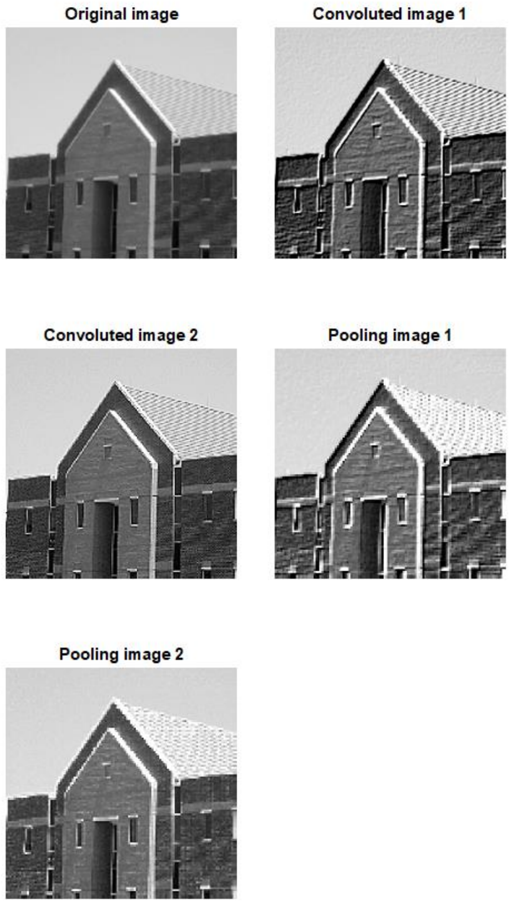


Fig. 24. Convolutional set cover and 1 filter each

In Figure 25 a complete convolution layer was made, with 2 sets of filters with 1 filter each, the kernels were entered by file, the software recognizes that two with-together filters are being used and creates 2 stages of sub-sampling, bias, and activation function.

The input of this layer was an image of 240x240 pixels, and the output is two images of 120x120 each.

Convolutional layers can be created with different settings and the program will connect each layer to the next layer.

The dynamic model of a manipulator robot contains in its mathematical structure parameters such as centers of gravity, masses, moments of inertia and friction coefficients [15]. These parameters are generally unknown; this is the case for most commercial robots where the manufacturer does not provide their nominal values. While there are control theory tools such as adaptive schemes and robust controllers that allow for errors in dynamic parameters, knowledge of these is crucial for most schemes based on the dynamic model of a robot.

The parametric identification problem has led to the derivation of several identification schemes that have become an attractive tool for determining the dynamic parameters of manipulating robots. especially when it is difficult to measure them directly. However, the non-linear nature of the dynamic model of manipulative robots makes the parametric identification task nontrivial.

System identification can be defined as the characterization of a dynamic system, observing its measurable behavior. When a priori information about the rules governing a system does not exist or is too complex, for example in electrocardiographic signals, identification techniques are used to construct a model only by observing the input output data. So, the system is called a black box, because the internal behavior is unknown.

In many physical systems, our knowledge of mechanical, chemical, or electrical laws allows us to formulate a model, which is the fundamental tool for studying the system, either analytically or through simulations. However, no matter how deep our physical perception, the parameters of any model are inaccurate.

There is an area of opportunity to introduce neural networks in this topic, in [16] use a Hopfield neural network for the estimation of parameters, in the context of the identification of a dynamic system. CNN could offer a solution to this problem by designing a neural network that would perform the parametric identification of a dynamic system, in a particular case that of a manipulative robot.

The dynamic model of a manipulator robot of n degrees of freedom is given by the Eq. (1) [15]:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f_f(\dot{q}, f_e), \qquad (1)$$

where:

- q Vector of generalized coordinates.
- $\dot{q}$ Vector of articular speeds.
- $\ddot{q}$ Vector of articular accelerations.
- $M(q)$ Inertial matrix.
- $C(q,\dot{q})$ Matrix of centripetal forces and Coriolis.
- $g(q)$ Vector of gravitational forces.
- $f_f(\dot{q}, f_e)$ Vector of friction pairs.

The number of parameters to be identified may be large if the manufacturer’s parameters are not available, or if it is not physically possible to measure them. The proposed platform allows to create a neural network for this purpose with the ease of being modified in the process for its adjustment or to increase the number of parameters, if at first you want to ignore the friction pair vector and then incorporate it the software allows you to do this without recreating the neural network from scratch.

5 Conclusions

The analysis of a neural network convolution allowed identifying the repetitive processes that the assistant can perform. The proposed method of creating an image as a convolutional network input using user-provided data allows a solution if the user does not have a created image and only has a data set such as an electrical signal.

The proposed platform can significantly reduce the development time of a neural network and helps to avoid code error when you want to adjust some parameter or configuration of the network.

This platform is aimed at researchers and scientists who wish to experiment with deep learning, opening it up to users who prefer GUI-based interaction and want to minimize the opportunity cost associated with software configuration, it is also sought that potential users who are not from the computational area approach the use of neural networks and that they can integrate it into their lines of research in other totally different areas.

The mentioned application exemplifies the approach with which this platform is designed, the objective of improving the design time of a CNN and the ease of modifying it when the application changes. Using the proposed image generation methodology, the applications of this platform are not only limited to the identification and classification of images, but CNN can also be performed for problems where numerical data are generated, whether mechanical, electrical, magnetic among others.

In the immediate future, the possibility of the platform generating the training stage will be implemented.

References

1. **Berzal, F. (2018).** Redes Neuronales & Deep Learning.
2. **Hamet, P., Tremblay, J. (2017).** Artificial intelligence in medicine. Metabolism: Clinical and Experimental, Vol. 69, pp. S36–S40.

3. **Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., et al. (2016).** TensorFlow: Large-scale machine learning on heterogeneous distributed systems. DOI: 10.48550/arXiv.1603.04467.
4. **Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T. (2014).** Caffe: Convolutional architecture for fast feature embedding. Proceedings of the ACM Conference on Multimedia. Association for Computing Machinery, pp. 675–678. DOI: .48550/arXiv.1408.5093.
5. **Bai, J. et al. (2019).** ONNX: Open neural network exchange. GitHub repository [Preprint]. GitHub.
6. **Frank, E. et al. (2016).** WEKA Workbench Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques.
7. **Paszke, A. et al. (2019).** PyTorch: An Imperative Style, High-Performance Deep Learning Library. In **Wallach, H. et al. (eds)** Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035.
8. **Seide, F., Agarwal, A. (2016).** CNTK: Microsoft's Open-Source Deep-Learning Toolkit. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: Association for Computing Machinery (KDD '16), pp. 2135.
9. **etBrains (2021).** Introducing Datalore Enterprise 2021.3: Database Connections, SQL Cells, Interactive Reports, and More. The JetBrains Datalore Blog.
10. **Berthold, M.R. et al. (2008).** KNIME: The Konstanz information miner. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 319–326.
11. **Klemm, S. et al. (2018).** Barista - a Graphical Tool for Designing and Training Deep Neural Networks. pp. 1–8.
12. **Sarvadevabhatla, R.K., Babu, R.V. (2015).** Expresso : A user-friendly GUI for designing, training and exploring convolutional neural networks.
13. **Octave (2021).** Octave for macOS.
14. **Octave (2021).** Octave for Debian systems.
15. **Reyes-Cortés, F. (2011).** Robótica. Control de Robots Manipuladores. ALFAOMEGA.
16. **Atencia, M., Joya, G., Sandoval, F. (2005).** Hopfield neural networks for parametric identification of dynamical systems. Neural Processing Letters, Vol. 21, No. 2, pp. 143–152.

Article received on 28/07/2021; accepted on 30/09/2021. Corresponding author is María Aurora Diozcora Vargas-Treviño.