# Implementation of library for teaching Convolutional Neural Networks

Jakub Stolarski

***Abstract*—The aim of this article is to make a review of the literature concerning Convolutional Neural Networks, their structure, potential uses and ways to train them in preparation for the upcoming task of Machine Learning library implementation in Julia. This review shall focus on presenting an abridged form of knowledge found in various scientific articles, rather than present an in depth analysis of Artificial Neural Networks and Deep Learning concepts.**

***Index Terms*—Convolutional Neural Networks, Artificial Neural Network, Machine Learning, Deep Learning**

## I. Intrudction and Literature Review

In recent years the concepts of Artificial Neural Networks (ANN) and Deep Learning (DL) in general were met with rapid development and many successes in fields of science and engineering [3]. The breakthrough of Machine Learning (ML) can be attributed to Neural Networks (NN) and their applicability towards solving a large class of difficult problems in computer science [7]. The main focus of this paper shall be an implementation of library in Julia language that is capable of teaching Convolutional Neural Networks (CNN), as such the general approaches to teaching a network, the structure of CNN and the capabilities of Julia language will be discussed in this section.

Although there are many approaches to teaching a NN, such as reinforcement learning and unsupervised learning, the focus of future project and thus also the focus of this paper shall be supervised learning. In this method prepared dataset, which contains input matrices and an appropriate annotations, is split into training set and testing set (and sometimes also into validation set). The training dataset is used to teach the network over numerous epochs and then the test set is used to check how well the model was trained. When it comes to teaching the network, most traditional algorithms rely on computation of gradients and Hessinans of an objective function [2]. Gradients and Hessians require the evaluation of derivatives, implementations of which is not a straightforward task [2]. In regards of differentiation methods: manually working out derivatives is the most time consuming method of derivation and the chances of miscalculations grow with the complexity of the equations, the method called numerical differentiation allows for simple implementation in code but is highly inaccurate and scales poorly with use of gradients, therefore making it not useful in ML [2]. Another two methods of symbolic differentiation (SD) and automatic differentiation (AD) address the weaknesses of other methods, but SD is largely believed to be riddled with problem of too complex expressions [2], although there are some who believe that both AD and SD are equivalent in their capabilities and that the ”expression swell” is not connected to the method itself [4]. Works of Soeren [4] and Innes et al. [7] focus on SD and AD respectively, whlie the latter also shows the way of implementing AD in Julia language, also the paper of Baydin et al. [2] is worth mentioning as it delves into theory behind AD.

CNN can be considered the true breakthrough of NN and are now widely regarded as the basic structure for image processing [1], [3], [5]. CNN contain an input layer, an output layer and multiple hidden layers. Different hidden layers and different connections between them may be used depending on CNN architecture, but three examples are worth mentioning - convolutional layers, pooling layers, fully connected (FC) layers. Convolutional layers, as the name suggests, use the mathematical operation of convolution between it's input and special filter called kernel [1], [5], [8]. Pooling layers' main task is the sub-sampling of the feature maps received on their input and transform them using certain methods (usually max pooling or average pooling [1], [8]). Lastly, FC layers are commonly used at the end of CNN architecture, in this layer each neuron is connected to all input signals of the previous layer [8]. The works of Osowski [1] and Alzubadi et al. [8] provide an excellent and in depth analysis of the core concepts of CNN and DL. Works of Dhillon et al. [3] and Lecun et al. [9] cover the learning process of CNN, first of which shows more theoretical approach, while the latter provides an excellent example of practical use. Paper of Tabik and el. [5], with their superb work on teaching CNN classifications of hand-written symbols, shall be used as a reference to train a CNN model in a referenctial solution.

The languge in witch the library shall be implemented - Julia, is a modern, expressive, high-performance and relatively new languages. Julia's grammar is similar to that of Python or Matlab but unlike those languages it can achieve C/C++ levels of performance allowing programmers to write clear, high-level, generic and abstract code that produces fast, low-level machine code [6], [10]. One of the basic premises of this language is that all basic functionality must be possible to implement in Julia, which deals with headaches of two language problem which is present in other high-level languages. Gao et al. [6] provide an excellent analysis of the language and it's DL applications (the latter of which is also mentioned in [7]), Bezanson et al. [10] provide an excellent introduction to the language it's basic principles and best ways to implement code in it.

## References

[1] Osowski, S. (2018). Głebokie sieci neuronowe i ich zastosowania w eksploracji danych. Przeglad Telekomunikacyjny - Wiadomości Telekomunikacyjne, 2018, 112–121. https://doi.org/10.15199/59.2018.5.2

[2] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. Journal of Machine Learning Research, 18.

[3] Dhillon, A., Verma, G.K. Convolutional neural network: a review of models, methodologies and applications to object detection. Prog Artif Intell 9, 85–112 (2020). https://doi.org/10.1007/s13748-019-00203-0

[4] Laue, Soeren. (2019). On the Equivalence of Forward Mode Automatic Differentiation and Symbolic Differentiation.

[5] Tabik, Siham & Peralta, Daniel & Herrera-Poyatos, Andrés & Herrera, Francisco. (2017). A snapshot of image Pre-Processing for convolutional neural networks: Case study of MNIST. International Journal of Computational Intelligence Systems. 10. 555. 10.2991/ijcis.2017.10.1.38.

[6] Gao, Kaifeng & Mei, Gang & Piccialli, Francesco & Cuomo, Salvatore & Tu, Jingzhi & Huo, Zenan. (2020). Julia Language in Machine Learning: Algorithms, Applications, and Open Issues. Computer Science Review. 100254-100259. 10.1016/j.cosrev.2020.100254.

[7] Innes, Mike & Edelman, Alan & Fischer, Keno & Rackauckus, Chris & Saba, Elliot & Shah, Viral & Tebbutt, Will. (2019). Zygote: A Differentiable Programming System to Bridge Machine Learning and Scientific Computing.

[8] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[9] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, ”Gradient-based learning applied to document recognition,” in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[10] Julia: A Fresh Approach to Numerical Computing. Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah. (2017) SIAM Review, 59: 65–98. doi: 10.1137/141000671.