A Neural Basis for the Implementation of Deep Learning and Artificial Intelligence

Alan F. Smeaton

Insight Centre for Data Analytics,
Dublin City University, Dublin 9, Ireland
{alan.smeaton}@dcu.ie

Abstract. One of the mathematical cornerstones of modern data analytics is machine learning whereby we automatically learn subtle patterns which may be hidden in training data, we associate those patterns with outcomes and we apply these patterns to new and unseen data and make predictions about as yet unseen outcomes. This form of data analytics allows us to bring value to the huge volumes of data that is collected from people, from the environment, from commerce, from online activities, from scientific experiments, from many other sources. The mathematical basis for this form of machine learning has led to tools like Support Vector Machines which have shown moderate effectiveness and good efficiency in their implementation. Recently, however, these have been usurped by the emergence of deep learning based on convolutional neural networks. In this presentation we will examine the basis for why such deep networks are remarkably successful and accurate, their similarity to ways in which the human brain is organised, and the challenges of implementing such deep networks on conventional computer architectures.

Keywords: Deep learning, neural computing, neural networks

1 Introduction

1.1 A Brief History of Computing

The short history of the development of computing is a mostly linear progression of the same basic principle of what makes a computer. Starting with the early thoughts on stored program computers by members of the team developing the ENIAC computer at the end of WW II and then leading on to the work of Alan Turing, this basic principle is about a device that stores a sequence of instructions to be executed in sequence, one after the other. We have seen major technological developments like the transistor in 1947, the integrated circuit in 1959, different forms of magnetic memory for storing instructions and data, and on the software side we have seen mainframes evolve into personal computers and now even wearable technology. The common characteristic across all these developments is the stored program model for computers proposed by John von Neumann in 1952. Here, a central processing unit retrieves an instruction from memory, decodes it, executes it and then moves on to fetch and then execute the next instruction unless directed to fetch the next instruction from somewhere else, as shown in Figure 1. This von Neumann architecture has not only carried computing since the advent of digital computation 60 or 70 years ago, but it is the basis for almost all kinds of computing that we perform today.

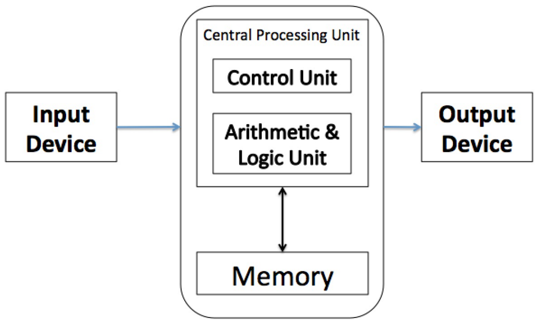

Fig. 1: The traditional von Neumann architecture for computers

2 Artificial Intelligence

2.1 Artificial Intelligence Emerges

Even since those early days, the question has been asked of whether computers could perform an artificial form of intelligence or execute tasks at the human level of performance. To do this we need to mimic the human brain, which although making up on 2% of our body weight, is made up of more than 80 billion neurons connected by trillions of connections or synapses. The brain is responsible for our executive functions like breathing, digestion, heart pumping, as well as even more complex tasks like planning, reasoning, and abstract thought. It achieves this using an architecture of a huge number (+80B) of simple, connected processors.

This architecture is good for solving complex problems like vision, and learning. The unit components of this architecture are neurons or simple perceptrons. Each neuron is on average connected to about 10,000 other neurons and neurons communicate by sending signals across these synapses. They operate by receiving a number of input signals from the multiple input connections or channels that they each have, aggregating the input signals in some way, called an activation function, and then based on the values of the input signals, they generate an output signal which is passed on to another neuron to which it is connected. While each neuron is very simple, it is the huge number of them, and their connections, that allow the overall system to address complex problem-solving, the kind that we call artificial intelligence.

So how do we implement artificial intelligence tasks, emulating the complexity of the human brain, on the von Neumann architecture which is so prevalent today ? Well clearly the von Neumann architecture with its sequential processing of instructions and in-built lack of parallelism, is clearly not a good fit for implementing neural computing. In the late 1980s we saw the first attempts at implementing a neural architecture directly in hardware through the development of the Connection Machine by the Thinking Machines Corp. While a faithful replication of neural information processing, the problem with the Connection Machine was that it only had some thousands of nodes with connections which was not enough to simulate the kind of intelligence at the level we seek for human problem-solving, so it was used for simple, massive parallelism applications like searching.

2.2 The Artificial Intelligence Winter

Around the time of the Connection Machine we were entering into the "AI Winter" with inflated expectations around AI leading to disappointment in the actual realization and a very deep "trough of disillusionment" as found in the classic Gartner curve. Research into neural networks — computational implementations of the brains structure — continued but at a slow rate of progress and forming a bit of an AI backwater. Meanwhile hand-crafted, rules-driven artificial intelligence research continued through the 1970s, 80s, 90s, and even 00s, with progress in AI applications in areas like speech, machine translation, computer visions and expert systems, for a while anyway.

The area of neural networks, while not hugely popular or attracting much research interest or funding, did continue to develop during this period and developed new configurations of neural networks like multi-layer perceptrons, convolutional neural networks [5], spiking neural networks [6], and more.

The simple configuration of a neuron is shown in Figure 2 where on the left side we show a simple neuron with 2 input connections, though there can be many more. The neuron accepts its two input signals, performs some operation on the signal values to generate an output signal which is sends. Neural networks are generally configured to form feed forward networks, operating on a set of inputs and generating one or more outputs. A simple feed-forward network is shown on the right in Figure 2 with 2 inputs,

Neural network architectures are much more complex that that shown in Figure 2 because they are so numerous. They usually have a number of hidden layers, an input layer, and an output layer. Adjacent layers are usually fully connected where every neuron in one layer is connected to every neuron in another layer. Layers can also have loops back to earlier layers, called recurrent neural networks, increasing both their complexity and the complexity of their problem-solving capabilities [11].

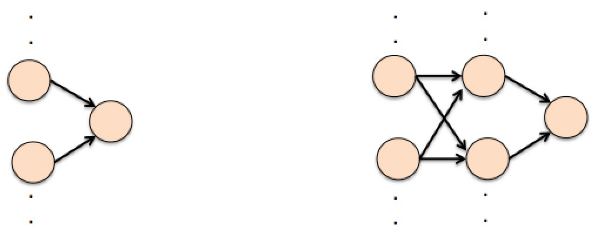

Fig. 2: Architecture components in neural networks: simple neuron (left) and feed forward network (right)

2.3 The Growth of Machine Learning

Meanwhile, as neural network architectures were developing, in other parts of the artificial intelligence area, *machine learning* developed as an AI tool and its development hinged on a background of mathematics and statistics rather than any idea of emulating human neural processing. Machine learning evolved slowly over these decades. It was nourished by the increasing availability of huge volumes of data from sources like internet searching, social media, online transactions, and others. One application area which pushed this development was computer vision and in this field we were able to use machine learning to train classifiers to identify objects and concepts appearing in images. To do this we used techniques like decision tree learning, random forests, genetic programming and most especially, support vector machines. The basic approach here was to use a large number of positive and negative examples, to extract low-level image features like shape, colour and texture, and let the computer figure out (or learn) how to classify new and unseen examples as to the presence or absence of these objects and concepts.

Up to the early and mid-point of the current decade, this worked OK and there was slow and incremental progress year-on-year but the techniques were not mature or accurate enough to be used in commercial or real applications; we had more work to do. At the same time, we also saw machine learning being used elsewhere in recommender systems for example, and this increased the popularity and visibility of machine learning as an AI technique. This standard or "classic" machine learning has several advantages including the fact that a decision or recommendation can be explained by examining the features or characteristics of each case, or recommendation. We can also rate the relative importance of each of the features or axes which gives us unique insights into the problem we are trying to solve. However, a downside here is that we have to do lots of feature engineering to define the axes of the problem space and there are no theoretical underpinnings to this and so it is a form of a black art.

3 Convolutional Neural Networks

3.1 What is a Convolutional Neural Network ?

Convolutional Neural Networks (CNNs) are a particularly complex architecture which have been around for more than 20 years but first really got noticed in [9]. Also known as a deep convolutional networks, they were developed to address image processing applications because their design allows an entire image to be fed into the network, so in theory there are a huge number of inputs to the network. The typical application for which CNNs were developed were to injest an entire image and to classify it as containing objects or concepts like "cat" or "dog" or "airplane" or "outdoor" or anything else that can be recognised visually.

However, rather than have an input node for each pixel in an image, which would lose whatever locality and proximity gives rise to us recognising cats, and dogs and airplanes and whatever else, CNNs create an input layer consisting of a window, a kind of sub-image, which slides across and scans the full image. This can be seen in Figure 3 where the input image is 32×32 pixels and a sliding window of $5 \times 5 \times 3$ corresponding to 5×5 pixels and 3 colours, and this window slides across the image, one pixel at a time, generating local context information to help recognise objects and concepts. This input is fed through convolutional layers where each node is not connected to every other one but to its local neighbourhood, and this structure is repeated a number of times. This is descried in more detail in [9].

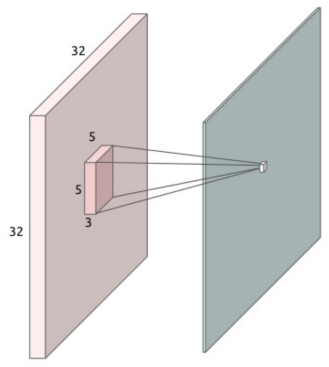

Fig. 3: Convolutional Neural Network showing sliding window in input layer

The idea of having such convolutions in a multi-layer network really gained notoriety in 2012, when Krizhevsky and his colleagues submitted a run to the ImageNet large scale visual recognition challenge based on an implementation of a convolutional neural network, and the performance of the submission was a huge improvement on the then state-of-the-art [8]. The technique was based on an architecture consisting of multiple layers of neurons, implemented using graphics processing units (GPUs) which allowed a huge increase in the number of nodes, and connections in the neural network. A schematic for such a networ is shown in Figure 4.

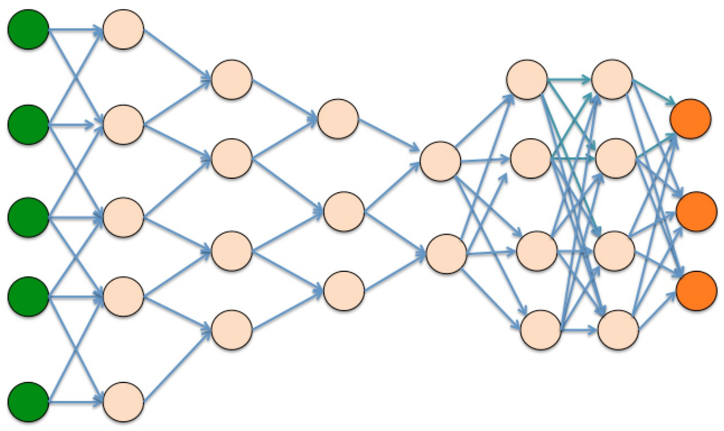

Fig. 4: Architecture of a Convolutional Neural Network

Convolutional Neural Networks are end-to-end solutions in which both the feature extraction covering features like colours, shapes and textures, and the classifier training are performed at once. These "deep features" turn out to be significantly more efficient than the classical ones which heretofore had been manually engineered, even when used with classical machine learning for classifier training.

Once a model for learning a concept is built, it can be packaged and easily run in a hosted environment and this has led to a huge uptake of this "deep learning" for computer vision problems. Google, Facebook and other major software companies now use deep learning as part of their commodity offerings for image tagging, video captioning, and other computer vision problems. To illustrate, many of the participants in TRECVid for concept detection, video captioning and other vision processing (as opposed to video search), now use CNN approaches [1].

3.2 Applications for CNNs

The use of deep learning via CNNs is now almost everywhere and being used for things like speech processing [5], circuit design [12], language processing [7], machine translation [2, 10] and others. These are nicely summarised by Le Cun *et al.* in [10]. Other applications in other areas include automative specifically self-driving cars, in banking it includes evaluating credit applications and evaluation of mortgage applications, layout of integrated circuits, predicting currency prices, predicting faults during a manufacturing process, analysis of EEG, ECG and fMRI signals, route and movement planning in robotics, vehicle and route scheduling in transport, and many more. Deng [4] shows this even broader range of applications for CNNs and deep learning which are showing some success, but we don't know whether this is only because of parallelism and whether they actually need a neural architecture which uses layers, or other even more complex structures like those described in [3].

3.3 Implementing Neural Networks

Implementing convolutional neural networks and similar deep learning architectures is definitely not suitable for a von Neumann architecture so for now, many implementations use graphical processing units (GPUs) which offer massive parallelism at a very cheap rate designed, as they are, for supporting graphics processing for applications like gaming.

For more longer-term implementations, we need to design new hardware chips to implement this new architecture, or rather this old architecture which has recently become popular. Intel are developing deep learning chips code-named Lake Crest and Knights Crest, while Samsung are targeting chips for handset devices which allow deep learning on devices. Other companies like Movidius, now part of Intel, are specialising in computer vision using deep learning on silicon. Recognising the emergent importance of deep learning and the computational expense this would require for applications like speech and vision, Google designed and built a new chip from scratch called the Tensor Processing Unit which implements deep learning on a neural architecture which is 30 times faster than on a von Neumann architecture.

4 Conclusions

Since the early days of computing we are used to having people writing algorithms, encoding these as programs which we store and run on von Neumann type architectures. Now, with the emergence of data-driven, AI-based technologies we have data which we use to build and train networks which emulate neural networks, and we develop models to solve problems like classification and recognition, which we store and run. This represents the greatest single shift in computing, notwithstanding the major technological contributions of things like the transistor, integrated circuit or magnetic memory storage. Understanding this shift, maximizing its impact and using it for the most appropriate kinds of application are things we hope we can do correctly.

Acknowledgements

This work is supported by Science Foundation Ireland under grant number 12/RC/2289.

References

1. G. Awad, J. Fiscus, D. Joy, M. Michel, A. F. Smeaton, W. Kraaij, G. Quénot, M. Eskevich, R. Aly, R. Ordelman, et al. Trecvid 2016: Evaluating video search, video event detection, localization, and hyperlinking. In *Proceedings of TRECVID*, volume 2016, 2016.
2. K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
3. H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan. *Neural Network Design*. Martin Hagan, USA, 2nd edition, 2014.
4. L. Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3, 2014.
5. L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, May 2013.
6. S. Ghosh-Dastidar and H. Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.
7. N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.
8. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
9. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
10. I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
11. R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.
12. C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 161–170. ACM, 2015.