Technological Prospects in Artificial Neural Networks and Image Processing

Conor Lorsung

***Abstract*—As artificial neural networks grow, so does their time and computational complexity. We will examine a general time complexity for these networks, and how it may be mitigated with parallel computing. We will also look at the architectural advancements in central processing units and graphical processing units and their effect on neural networks. Finally we will cover different techniques for constructing smarter, more efficient networks. This includes: constructive artificial neural networks, biological contributions to research, and increased dimensionality of neural networks.**

I. Introduction

Artificial Neural Networks (ANNs) are a powerful virtual tool with applications in pattern recognition. Examples of such applications are: character recognition, medical landmark determination, and image saliency mapping. Thus far, ANNs have given been used to generate better voice recognition, classify objects in images (i.e. finding human faces in a picture or video), and more. [2] We will examine some of the current bottlenecks in the construction of ANNs and explain where they will continue to develop. The prominent areas of research we have identified include [3], [9]: parallel processing, Constructive Neural Networks, and ANN architectural expansion. For each area, we will look at the technological viability of growth and the direction in which we expect these developments to pursue.

II. Parallel Computing

Parallel computation is defined as the concurrent execution of processes or calculations. Typically this includes the use of multiple Central Processing Units (CPUs) working together on a single task for quicker evaluation. Additionally, Graphics Processing Units (GPUs) may be used in parallel computing. Depending on the nature of the given process, CPUs and GPUs will perform differently due to their hardware architecture. ANNs rely on the calculation of many vectors, given that the artificial neuron input layers are typically modeled as a vector. This means architectures better suited to matrix multiplication will perform faster than those that are not. We will cover specific cases where each unit excels, the growth of unit technology, and the implications this has for ANNs. First, it is necessary to understand why this is needed.

A. ANN Time Complexity

The following time complexity is an estimation of ANN weight convergence performance [7]:

$$3\sum_{j<i}|w_{ij}| = O(n^2 \cdot \max_{i,j}|w_{ij}|) \tag{1}$$

where w is the weight value at neuron i of layer j. In this equation, n is the number of bits needed to represent the input. This complexity represents the total number of neuron state changes using an asynchronous update rule. Asynchronous in this regard means each neuron is updated in some sorted order. Parallel computing seeks to minimize this complexity by adding synchrony to the computation. This is achieved by using multiple processing units.

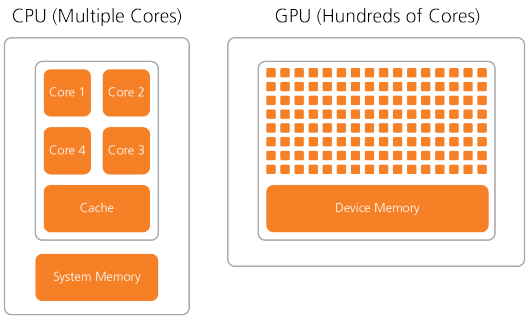

Fig. 1. The figure above illustrates the basic architectural differences between CPUs and GPUs. As shown, CPUs have fewer cores compared to GPUs. This is due to their different use cases: general processing (no strict need for synchrony) and matrix processing (largely asynchronous) respectively.

B. General Unit Architecture

Parallelism makes use of the cores in a processing unit and their threading capability. That is, each core may run a separate program since they contain individual processing threads. Taking this into account, we may delegate almost as many tasks to the processing unit as it has cores. Using this technique, it becomes apparent why parallel computing is typically faster than single core computing. In the case of ANNs and their need for many computations, parallelism becomes ideal since it allows for greater overall computational throughput. One product of increasing core counts is quicker computing.

C. The Central Processing Unit

CPUs are the units in a computer system that carry out arithmetic, logic, and input/output of a computer program. The widely used format of these units today is the microprocessor: oftentimes a single-die with many transistors to perform logical tasks. According to Moore's Law, the number

of semiconductors that can fit onto a single computer chip doubles roughly every two years. [6] With this idea in mind, it is easy to see that computers have become very fast.

*1) **Applications to ANNs:*** The CPU has a smaller number of cores compared to the GPU, but they are generally much faster. [5] This allows for the CPU to process calculations at each neuron quickly, but it is limited by the number of cores the CPU has. This is an example of low throughput. Despite the limited number of cores, Moore's Law still exists and we have seen a growth in the number of both cores and semiconductors in CPUs. [6]

*2) **Future Analysis:*** If Moore's Law persists, we expect to see a growth in the number of high-powered cores and semiconductors in CPU architecture. Despite CPUs not being explicitly engineered for matrix processing, they remain a staple jack-of-all-trades unit that can quickly process smaller tasks. This will provide quicker training times for ANNs and the ability to upscale the amount of neurons and hidden layers.

D. The Graphics Processing Unit

GPUs are good at parallel computing since they have a large number of cores relative to the CPU. This is shown in Fig. 1 As GPUs have many cores, they can handle larger amounts of data than the CPU. Additionally, they can execute tasks synchronously; meaning the GPU will have a large throughput.

*1) **Applications to ANNs:*** Given that the GPU has many cores, it becomes an ideal candidate for matrix multiplication. Generally, this is the underlying purpose of the unit. For example, rasterizing images is done through matrix transformations and arithmetic. Given that ANNs are composed of vectors and matrices, a GPU can be used to evaluate large output of the network quicker than a CPU by computing in parallel.

*2) **Future Analysis:*** Moore's Law also affects GPU architecture. With an increasing number of semiconductors and cores, GPU performance metrics are expected to rise. In addition to the architectural improvements demonstrated in Fig. 2, software that streamlines these processes is being developed. One example of this is the consumer and academic software built by NVIDIA for GPU computation and parallelization. [1] We expect that architecture will continue to advance at the rates is has, thus providing improvements in ANN size and learning, specifically relating to image processing. [6]

III. Evolutionary ANNs

Evolutionary ANNs are artificial neural networks that learn the data they are processing and form their structure around it. One example of this is the Constructive Artificial Neural Network (CoNN). These networks make use of a *Constructive Algorithm* that generates additional nodes, layers, and connections as needed to best fit the training data. [9] There are six unique motivators for using CoNNs. These are outlined in [9]:

1) Flexibility of exploring the space of neural network topologies
2) Potential for matching the intrinsic complexity of the learning task
3) Estimation of expected case complexity of the learning task

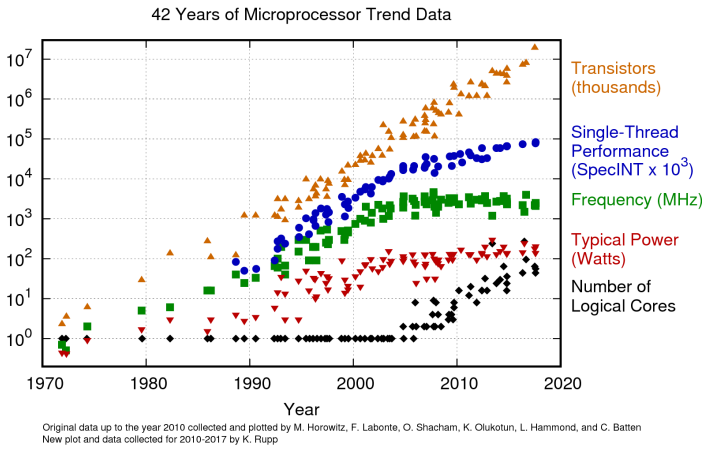

Fig. 2. This figure shows the trends in microprocessor data. It is apparent that Moore's Law (coded by the Transistors category) is still effective. Additionally, we can see an increase in unit cores; adding to the efficacy of parallel computing. Also significant is the single-thread performance, as cores execute programs on their threads. The figure indicates that parallel processing is more viable than ever, given an increase in thread performance and number of cores. We expect the trends to continue with the support of Moore's Law and historical data.

4) Tradeoffs among performance measures
5) Incorporation of prior knowledge
6) Lifelong learning

Considering the points above, we see how these networks can be useful in applications relating to image processing. Since CoNNs readily adapt to training data and further input data, they can be trained on a wide variety of inputs (within reason). As these systems are further researched, and expanded, we will gain a better understanding of their behavior. Given the current state of research into CoNNs [8], we postulate that generalized models will be developed. As processing power continues to grow (see sections II-C & II-D), we can expect these networks to compute, learn, and shape faster. This will give us structural information quicker, as it will become feasible to use larger and more complex networks.

IV. ANN Architectural Expansion

As ANNs grow, their computational time complexity does too. This is a problem, since the time complexity given in Eq. (1) grows roughly to the order of n^2. There are other ways to further develop ANNs besides creating more layers.

A. Biological Contributions

Using *coding schemes* to evaluate the transfer of data. These could be time-series evaluations of output and activation frequencies of neurons in the ANN. *Coding schemes* are currently used to evaluate biological brain performance. [4] By further examining the biological brain, we may uncover valuable information to better construct our virtual networks.

B. Specialized Hardware

One limitation of growing ANNs will be the need for specialized hardware. Neural networks that may find use with thousands of neurons will require large amounts of power

to operate through just one iteration. [4] Therefore it will be necessary to create power-efficient devices that can meet the memory, computational, and speed requirements of larger ANNs.

C. *Multidimensional Neural Networks*

While these only exist in theory currently, they would be an improvement upon the ANNs we have today. "The theory is based on the concept of generalizing one-dimensional logic gates to multi-dimensional logic gates and in a similar manner one-dimensional neural networks have been generalized to [Multidimensional Neural Networks] MDNN's." [4] The authors further explain that the MDNN receives input from the states of a hypercube, then figure out the stable states afterwards.

V. Conclusion

The ANNs of today have found many worthwhile uses. However, as we saw in sec. II, they are computationally cumbersome and rely heavily on matrix processing. Currently, CPU and GPU architecture is the fastest it has been. In addition to this, it also shows promise in continuing to grow. Aside from the speed of these units, the throughput of them is expected to grow as well. This is due to the parallel architecture of GPUs and the development of specialized parallel computing software. Outside of hardware, further developments in ANN structure will provide beneficial results. With generalized forms of CoNNs and the development of MDNNs, we expect ANNs to become better learners for provided tasks – whether it is character recognition or medical landmark identification. Not only will these provide enhanced learning, they may become persistent machines that can be trained for multiple purposes.

References

[1] About cuda. https://developer.nvidia.com/about-cuda.

[2] Robert D. Hof. Is artificial intelligence finally coming into its own?, Mar 2016. https://www.technologyreview.com/s/513696/deep-learning/.

[3] Shivani Joshi. Artificial intelligence: Prospects in artificial neural network (ann). *International Journal of Scientific & Engineering Research*, 7(12):427431, Dec 2016.

[4] Sachin Lakra, T. V. Prasad, and G. Ramakrishna. The future of neural networks. *CoRR*, abs/1209.4855, 2012.

[5] Christian de Looper. What's the difference between the cpu and gpu?, Feb 2017. https://www.pcmech.com/article/cpu-vs-gpu/.

[6] Chris A Mack. Fifty years of moore's law. *IEEE Transactions on Semiconductor Manufacturing*, 24(2):202207, May 2011.

[7] Pekka Orponen. Computational complexity of neural networks: A survey. 1, 05 2000.

[8] Beatriz Pérez-Sánchez, Oscar Fontenla-Romero, and Bertha Guijarro-Berdiñas. A review of adaptive online learning for artificial neural networks. *Artif. Intell. Rev.*, 49(2):281–299, February 2018.

[9] Sudhir Sharma and Pravin Chandra. Constructive neural networks: A review. 2, 12 2010.