Contents

[**1** **Background** 2](#_Toc70954072)

[**1.1** **Neural Networks** 2](#_Toc70954073)

[**1.2** **Convolutional Neural Networks** 3](#_Toc70954074)

[**1.3** **CNN Acceleration** 5](#_Toc70954075)

[**1.4** **Field-Programmable Arrays** 5](#_Toc70954076)

Table of Figures

[Figure 1: 2D convolution illustration 4](#_Toc70954120)

[Figure 2: Max pooling illustration 4](#_Toc70954121)

[Figure 3: ReLU function 5](#_Toc70954122)

[Figure 4: Softmax function 5](#_Toc70954123)

## **1 Background**

### **1.1 Neural Networks**

Neural networks are a family of computation architectures inspired by the biological nervous system. That system consists a large amount[[1]](#footnote-1) of neurons, which are electrically excitable cells, that are connected to each other in junctions named synapses[[2]](#footnote-2), forming a massive network. Each neuron receives input signals and produces output signals which branch out and connect to the other neurons. The synapses connecting the neurons influence the transfer of information from one neuron to the other by amplifying, attenuating, or inhibiting the signals transfer. Together, the billions of simple neurons form an incredibly complex interacting network which enables all the brain activity.

The basic building block of an Artificial Neural Network (ANN) is the artificial neuron. The artificial neuron receives several input signals from other neurons. These input signals are multiplied with weights to simulate the synaptic interaction. The weighed input signals are summed, biased, and fed into a non-linear activation function, which produces the neuron’s output signal. A neural network is formed by interconnecting many artificial neurons. Usually, the neurons are grouped into layers, and connections are only allowed between neurons of adjacent layers. Such layered connections are called Fully Connected layers.

When neural networks are employed for image-related tasks, their input usually consists of pixel data. Even for images with modest resolutions, the input consists of large amounts of elements (especially when dealing with multiple channel images, such as RGB). Large input data results in a need with amounts of neurons and connections that are a challenge for computing systems, and place a burden on the efficiency, both in time and in power, of analyzing images effectively in such neural networks.

### **1.2 Convolutional Neural Networks**

Convolutional Neural Networks (CNNs) are a special class of neural networks, suited for operation on 2D input data such as images. The idea behind the CNN is the locality of information in images, which means that important information in images  
can be captured from local neighborhood relations. Strong contrasts indicate edges, aligned edges result in lines, combined lines can result in circles and contours, circles can outline a wheel and multiple nearby wheels can point to the presence of a car. This locality of information in images is exploited in convolutional neural networks by adding new layers before the fully connected layers, namely convolutional layers, and pooling layers.

The purpose of convolutional layers (CONV) and pooling layers (POOL) is to reduce the spatial data of 2D images into smaller data sets, that contain the most distilled data from within the image. After that process is done, the refined data can be passed through fully connected layers (FC), which will now be much smaller and tailored for the purified information. A CNN may have multiple CONV and POOL layers.

* **Convolutional layer** – This layer preforms the mathematical 2D convolution. This operation is described by Eq. (1):

|  |  |
| --- | --- |
|  | (1) |

The operation basically slides filter of size , where each pixel in the filter has its own weight , over the 2D image, and computing the sum of the product of all overlapping pixels. The filter is also known as a kernel, and the convolution output is called a feature map. The operation is described in **Figure 1**.

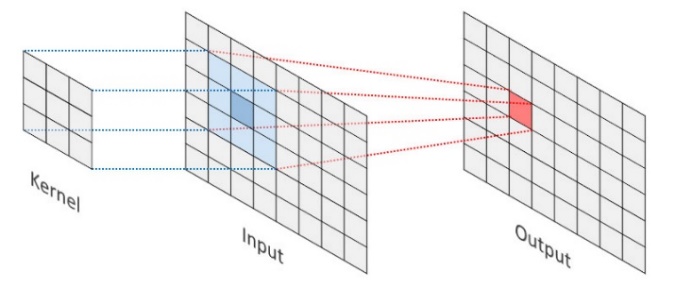


Figure 1: 2D convolution illustration

* **Pooling layer** – This layer reduces the spatial area of the given input, usually being the output of a convolutional layer. This is done by selecting the max value or calculating the average value from a 2D section. These are called Max-Pool and Avg-Pool, respectively. Max polling can be visualized in **Figure 2**.

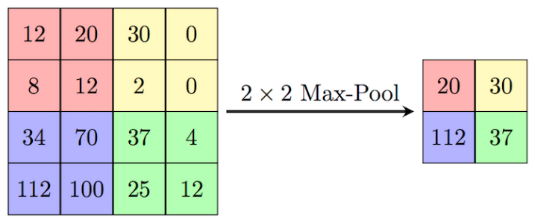


Figure 2: Max pooling illustration

* **Fully Connected layer** – The operation of FC layer can be expressed by Eq. (2). Input features are fully connected to an output feature vector by weights (). The and are the lengths of the input and output feature vector and the number of weights is . is the corresponding bias term of the th output feature vector. FC layers usually turn the input feature maps into a feature vector.

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

* **Activation Layer –** This layer introduces non-linearity to the CONV and FC outputs. The non-linearity increases the power of deepening the network. The most common activation functions used in CNNs are the Rectified Linear Unit (ReLU) function expressed in Eq. (3) and seen in **Figure 3**, and the SoftMax function expressed in Eq (4) and seen in **Figure 4.**

|  |  |  |  |
| --- | --- | --- | --- |
| (3) |  | (4) |  |
| Shape  Description automatically generated  Figure : ReLU function | | Chart  Description automatically generated  Figure : Softmax function | |

### **1.3 CNN Acceleration**

As CNNs achieve very high accuracy in image understanding and classification, they get embedded in more applications, some of which require tight real-time constraints. This leads to the need to accelerate the CNN calculations.

Most of the computations done in the CNN is in the convolutional layers, around 85-90%[[3]](#footnote-3). Since all layers: CONV, POOL and FC, basically deal with matrices, the software implementation of those layers consists of nested loops. This means all computations are done serially, and thus the time the network operates on a given output may be very long.

To reduce the computation time, and thus to accelerate the CNN performance, the different layers can be implemented in hardware. Since hardware can preform computations in parallel, this basically allows to expand the software loops so that a CONV single operation can be done in a single clock cycle.

Potential hardware platforms that can be used for accelerating the CNN are DSPs, GPUs, ASICs, and FPGAs. The advantages for using an FPGA are discussed in the following section.

### **1.4 Field-Programmable Arrays**

Field-Programmable Gate Arrays (FPGAs) are semiconductor devices consisting of a 2D array of configurable logic blocks which are connected via programmable interconnects. The interconnect can be thought of as a network of wire bundles running vertically and horizontally between the logic slices, with switchboxes at each intersection. Modern high-end FPGA generations feature hundreds of thousands of configurable logic blocks, and additionally include an abundance of hardened functional units which enable fast and efficient implementations of common functions. The logic blocks, the fixed-function units as well as the interconnect are programmed electronically by writing a configuration bitstream into the device. The configuration is typically held in SRAM memory, and the FPGAs can be reprogrammed many times.

The advantage of FPGA-based systems over traditional processor-based systems including GPUs, is the availability of freely programmable general-purpose logic blocks. These can be arranged into specialized accelerators for very specific tasks, resulting in improved processing speed, higher throughput, and power savings. This advantage comes at the price of reduced agility and increased complexity during the development, where the designer needs to carefully consider the available hardware resources and the efficient mapping of his algorithm onto the FPGA architecture. Further, some algorithmic problems do not map well onto the rigid block structures found on FPGAs. Compared to ASIC (Application-Specific Integrated Circuits) implementations, the FPGA advantages are re-programmability, that allows fast prototyping and short development cycles.

1. For example: a mouse brain has approx. 70 million neurons, human brain approx. 100 billion neurons [↑](#footnote-ref-1)
2. Mouse: approx. 1 trillion synapses, human: approx. 125 trillion synapses [↑](#footnote-ref-2)
3. link [↑](#footnote-ref-3)