Group Research Project 11: Phase 0

Introduction

Digital signal processing (DSP) is fundamental to modern technology. DSP algorithms are used in many industrial and safety-critical applications. Attempting to monitor and analyse continuous-time analog signals present in the environment using fundamentally discrete digital systems introduces many challenges, especially when the time predictability, low latency and high throughput of the system are crucial to the application behaviour. Hardware accelerators are commonly used in DSP to improve performance at specific tasks, especially those involving concurrency or SIMD operations. Hardware acceleration circuits may be integrated with larger, general purpose systems as coprocessors. Example hardware acceleration platforms include application specific integrated circuits (ASICs) and coprocessors, in this project implemented on a soft-core field programmable gate array (FPGA) [1].

An emergent area of work in the field of DSP is the use of machine learning and neural networks to perform feature extraction, pattern recognition, advanced heuristics, and other tasks with complex and dynamic demands. Convolutional neural networks (CNN) are used in audio and speech recognition, notable examples including the ubiquitous “Hey, Google” [2].

CNN inference is compute and memory intensive due to the large number of matrix operations and is typically unsuited for resource-constrained systems. Its operations largely employ the sequential SISD general purpose processor architecture. GPUs are commonly used in hardware acceleration of machine learning algorithms but have poor energy efficiency, high memory demands, and lack adaptability for use with irregular parallelism. Additionally, audio recognition CNNs must pre-process the audio signal to create a spectrogram for analysis (a spectrogram being a matrix representation of the magnitude of frequencies in a signal as it varies with time). Conversion from the time domain to the frequency domain requires computationally involved Fourier transforms [3,4].

In this project we will define the architecture of a heterogeneous multiprocessor system on chip (HMPSoC) for the application of processing audio/speech pattern recognition algorithms in edge inference. The system features a time-critical section designed to manage time-conscious hardware acceleration for key operations in CNN audio recognition inference. This acceleration may take the form of custom instructions called directly by the general purpose processor (GPP), or configurable automatic run-modes useful to our application. The HMPSoC will incorporate different types of processor cores, components and connection interfaces to support mixed-criticality systems. This includes a reactive and concurrent RISC processor, ReCOP, a general purpose processor (GPP), Nios II, and custom data processing application-specific cores for different CNN operations, coded as DP-ASPs.

Application Decomposition

A CNN is a system of layers of neurons connected in a feed-forward manner. An input propagates through the neural network and returns meaningful classification or prediction of some inputs. At a low level a CNN can be represented as a collection of matrices, and the operations on neurons as matrix operations. Some notable operations are detailed below.

DP-A: Spectrogram Operations

A spectrogram is used to represent the magnitude of frequencies in a signal as it varies over time, the signal being discrete and sampled at a fixed-time interval. The analysis involves the short-time Fourier transform, summating the Fourier transform of overlapping windows of a signal as it changes over time. This involves the application of a window function to a signal (vector multiplication) to extract the weighted samples, applying the Discrete Fourier Transform (DFT) to the window, converting the correlations from complex numbers to magnitudes, and then repeating for a given number of overlapping windows.

Spectrograms are used as an input to speech/audio recognition CNN to visualise audio-signals in 2D space, and so must be performed repeatedly on the signal of interest. Hardware acceleration is especially important as the time to calculate the DFT limits the sample/analysis rate, and thus the time-domain resolution and measurable frequency range.

Of interest to us is accelerating the DFT algorithm due to its reliance on sinusoid terms and large number of element-wise concurrent multiplications. It is important to reduce the DFT analysis time of each each sampled window to increase throughput (allowing for an increased sampling rate and a higher maximum measurable frequency) and reduce latency (decrease the interval/step-size between windows, improving spectrogram resolution in the time domain and responding to changes in the signal faster).

The DFT-ASP will perform DFT on an input vector of specified (2^N) length, where the vector elements are signed integers of a specified word bit-length, for a configurable range of frequencies, returning a vector of magnitudes for the frequencies.

DP-B Pooling and Batch Normalisation Layers

Pooling is done to extract features from the data while minimising the length and width of the input data. This is done by having a moving window and applying a function to each section. The output of the function for each section is then the corresponding pixel value in the new image. The size of the window determines the size of the output data, where:

$$Output\ Width\ =\ (Input\ Width\ -\ Filter\ Width)/Stride\ Length\ +\ 1\ Output\ Height\ =\ (Input\ Height\ -\ Filter\ Height)/Stride\ Length\ +\ 1$$

There are many functions that are used in pooling, such as max, average, median and sum. The application of these functions are in the name - max takes the maximum value within the window, average takes the average of the window, median takes the median, and sum sums all the values within the window. However, max and average are the most commonly used [5]. A selection of

methods will be provided in the ASP configuration to allow the user to see the effect on different functions for a given input.

Max

Max takes the maximum value within the window and applies it as the value for the corresponding pixel in the new image. However with max pooling, if there is no hard vector quantisation (no obvious feature within the input) max pooling will do poorly[6]. Similarly, high frequency audio could mask out the audio features we are looking for if our desired frequency is similar to unwanted noise frequency, resulting in high amplitude high frequency waves.

Average

Averaging takes the average of all the values within the window and applies it as the value for the corresponding pixel in the new image. Averaging has been shown to work well when small windows are used and when whole-image-pooling is not used [6].

Pooling Cardinalities

There are also different pooling cardinalities such as whole image pooling and level spatial pyramids that determine what parts of the image are pooled. This plays an important role in the effectiveness of the pooling functions and will need to be looked into to either provide configurability or optimise for audio.

Another important factor is what value is used to determine the importance of a particular segment. For audio recognition, this would be the amplitude of each frequency within an input spectrogram.

Batch Normalisation

Another part this ASP will look into is Batch Normalisation. When training neural networks, many hyperparameters for training such as batch size and number of training cycles, referred to as epochs, are used. This has led to improved performance tuning for neural networks. However, the problem with changing these parameters is that once one parameter is changed to improve a single layers performance, these changes are propagated through the rest of the network, amplifying the effect of these changes. Thus, the more layers a neural network has, the harder it is to tune these parameters for optimal efficiency. This is known as the internal covariate shift. Batch Normalisation reduces the internal covariate shift by adding a layer that normalises the output of the previous layer, to the input of the next layer. This has been shown to reduce the training steps by 93% while also receiving higher accuracy.

DP-C: Convolution Operation and Configurable Activation Data Processor

CNN performs discrete convolution over the spatial dimension of 2D data. In most meaningful cases, such operations are characteristically repetitive in processing the input and hidden layers. Such operations can be reimagined mathematically to simplify computation expenses, such as the Winograd transform which reduces multiplication operations in favour of using large coefficients. Another is to flatten the input and rearrange the kernel weights into a sparse circulant matrix, to achieve the same convolution with matrix multiplication. All of the above have the opportunity for paralliation in hardware through dedicated circuitry.

Most notably are the Multiply and Accumulate (MAC) arithmetic operations for computing dot products, implemented by a set of parallelised multipliers and adders. To reduce the need for multipliers, a shifter circuit can produce partial products that can be later summarised. The after

method has the benefit of supporting weights of different bit-widths over different configurations which is quite valuable for this project [7].

In the context of this project, basic configurable aspects of CNN (apart from the input) that are available to be integration to the ASP to be configurable includes:

- **Kernel Size**
  - **Kernel Weights**
  - **No. of Kernels (depth among a single stage) and/or channels**
- **Stride:** Shrinks layer (within a similar layer).
- **Padding:** Prevents shrinking (within a similar layer).

Activation functions are deliberately applied to convoluted outputs to introduce non-linear properties to the neural network's approximation of a function. ReLU activation remains the most renowned activation function courtesy to [8,9]. While the ReLU operation is simple and mitigates problems like the vanishing gradients, it is but only one option suited for a particular application. For example, to predict values that are larger than 1, tanh or sigmoid are not suitable to be used in the output layer, instead, ReLU can be used. On the other hand, if the output values have to be in the range (0,1) or (-1, 1) then ReLU is not a good choice, and sigmoid or tanh are more advantageous in this regard. Hence, it may prove to be useful to perform parallelisation of these functions and offer the option to config for an activation function. Moreso, if the DP is able to adapt and reconfig dynamically based on the output values.

Scope
This processor core will offer the following functionalities in the following priorities

- Custom instruction call convolution
- Padding, stride, depth config option
- ReLU Activation parallelisation

The design plans to follow the steps in using barrel shifters to perform a parallelised version of the matrix inner product which achieves the same result as convolution. This implementation gives us the option to parallelise the same convolution operation for different channels. However, the uses of sub-integers still require some investigation.

The integration of a configurable function with the core is provisional and not a part of the fundamental scope due to time-limitations. Justified by the popularity of the ReLU activation and its wide usage, we chose to limit ourselves to the feasible and satisfactory integration of only the ReLU function.

DP-D Classification Processor for Dense Layers
After the convolution and pooling operations, the signal reaches the classification stage, which consists of fully connected layers, also known as dense layers. A fully connected layer consists of an input stage, which is taken as the output from the final pooling operation, where all the neurons are flattened into a single vector. Weights are then applied to the neurons in the flattened vector. These activation neurons have full connected to all neurons in an output layer. Backpropagation takes place throughout to determine the most accurate weights for classification. Neurons in a fully connected layer have full connections to all activations in the previous layer. The connections between neurons are trainable weight parameters that are trained to associate features with a label. Before the network

reaches the output layer, a last activation function is applied. The number of neurons in the softmax function must be equal to the number of output categories. The operations involved with this classification stage of the network to be implemented on a DP-ASP are as follows:

- Flattening
    - Converts an NxNxN size matrix to a 1D vector of size NxNxN
- Fully connect/dense layer
    - Takes an the flattened input vector x
    - Applies a weight matrix W with dimensions [prev_layer, next_layer]
    - Weights learnable parameters, updated through propagation, however initial value for weights can be set
    - Bias vector b, which is a set of weights that require no input, for pre-output layers
- Activation function
    - Sigmoid is used for binary classification, and softmax is used for multi-class classification
    - Activation function represented by g

Hence the fully connected layer can be represented by the equation: $g\ (Wx + \ b)$. its The activation function is performed on its biases summed with the dot-product of its inputs and weights. The softmax function is performed on the output neurons of the fully connected layer. With this function, probability values of the different classes are processed and made to add up to 1, and hence the output of a network is normalised to a probability distribution over predicted output classes.

Architecture

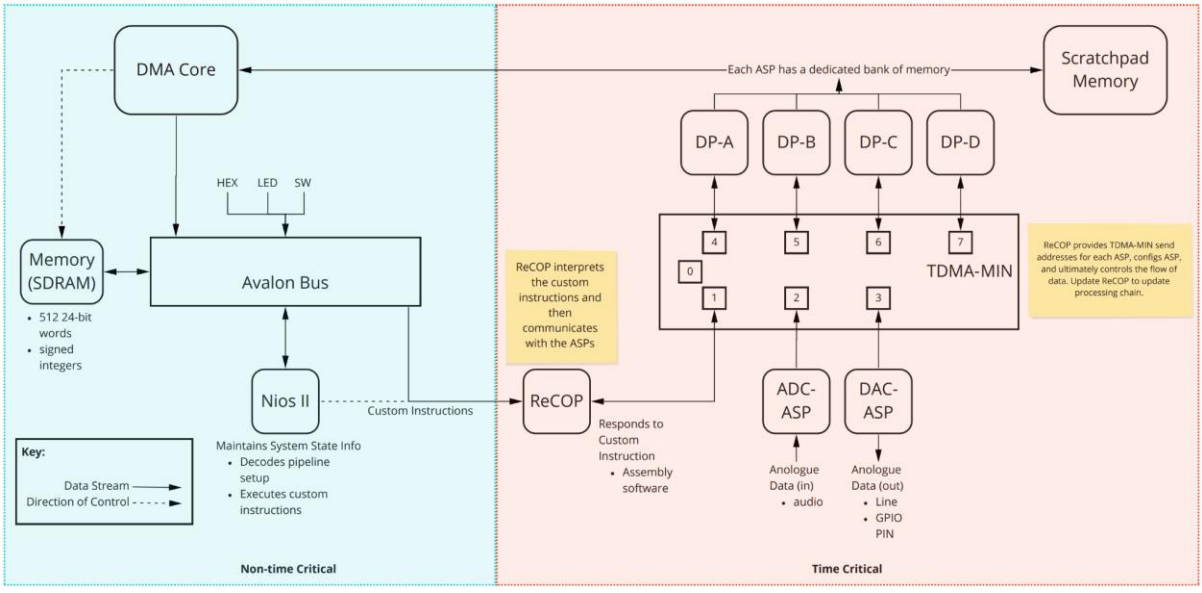

Fig. 1. High-level system architecture diagram

The HMPSoC allows connection of a time predictable network-on-chip region to the general purpose processor (GPP), Nios II, and the system as a whole. This time predictable region comprises our ASP designs interconnected with a TDMA-MIN NoC. The ReCOP is responsible for coordinating the execution of the ASPs and interfacing with the GPP. We intend for the ASPs to be largely autonomous, the ReCOP acting as a ‘master’ control unit and performing high level coordination,

using the ASPs and NoC as its datapath. The GPP may interact with ASPs directly in the case of some custom instructions.

In our speech/audio recognition CNN application the GPP will interact at a high level with the time predictable region. It will have the ability to reconfigure the processing pipeline (for a set of configurations supported by the ReCOP), read output data, and .

In the case of 'hotword' recognition networks [2] the GPP would perform unrelated tasks while waiting for an interruption or specific classification from the CNN, indicating a high degree of decoupling of control of the time predictable section from the GPP.

System Architecture Interfaces

Time Critical

TDMA-MIN

The TDMA-MIN NoC is responsible for communication between our ASPs and the ReCOP within the time critical section due to its time-determinate message passing and guaranteed bounded latency. The TDMA-MIN will have up to 8 ports, connecting to the different ASP as mentioned in the process decomposition,

Communications may have the following purposes:

- ASP and/or ReCOP Configuration
    - ASP-functionality-specific configuration signals (parameters).
    - Assigning input or output addresses for NoC to control data flow.
- Data
    - Generic data packets, their contents determined contextually or with ASP-specific encoding.
- Custom Instructions
    - addr_A
    - addr_B

Custom Instructions

The underlying fabric of custom instructions will communicate with the ReCOP processor through the Avalon bus. As a HMPSoC, our system will aim to be cycle accurate, but we must be ready to adapt to the situation where an operation might not be single cycled. The custom instructions are multi-cycle and extended, wired to deliver the data to the ReCOP processor and waits for the returned results plus done signal. The extension will be able to differentiate the set instructions as a selector to inform the ReCOP processor of the initiated instruction. The ReCOP processor will be equipped with its own assembly-specified software algorithms to perform the operations on the NoC.

ReCOP

As part of the critical interfaces, a reactive and concurrency processor (ReCOP), a general purpose RISC core is connected to a port of the TDMA-MIN. The ReCOP combines the data-processing functionalities of a traditional processor (ARM, NiOS, MIPS etc) with a reactive co-processor, for execution of control and reactive applications. The ReCOP is optimised for applications that receive events from the environment at unpredictable times and require output to be generated as a reaction to such events. The ReCOP is time-predictable due to its simple ISA, design, pipe and no caches.

Complex data processing is carried out by external data processors outside the ReCOP, relayed back to the ReCOP for signal values. The ReCOP architecture, inspired by ReMIC design, consists of a number of register components that executes control, or communicates with other data processors or the environment.

Non time-critical

Avalon Bus

The avalon bus fabric provides the primary system connection for the non-critical region. An avalon bus interface is responsible for communicating between the non-critical and critical sections of the HMPSoc, connecting the non-critical general purpose NIOS II processors with the ReCOP. The interfaces between Nios II GPP will not be elaborated here, but the interface for ReCOp and ASP-Memory relationship shall be elaborated.

All Avalon bus interfaces shall be off the specification of the Avalon-MM interface, including the DMA core.

Nios II

A Nios II processor will be the general purpose processor in the non-critical section. It will be connected to the avalon bus with double direction flow of data. The Nios will maintain system state information while decoding pipeline setup. It will also have a level of control over the ReCOP, by executing and sending custom instructions. The Nios processor will also support non-critical functions such as preparation of data, change of code of critical part and debugging. A SDRAM for the NIOS will also be connected to the avalon bus.

Architecture Scope

Parameters

For a properly configurable ASP we should consider supporting a range of value formats, including various word lengths (for example, 8/12/16/24/32+ bit words), vector compression (e.g., a 24-bit packet containing three 8-bit words), signed and unsigned integers, and even potentially fixed-point integer operations (currently out of scope).

Compression techniques are out of scope but of interest to CNN. For example, quantizing to 8-bit fixed-point integers can be achieved without significant loss in inference accuracy compared to FP32 models [7]. Quantisation can improve memory throughput, alleviate on-chip memory requirements and save circuit space. Other compression techniques include sparse matrix representation, quantisation and lossy interpolation compression, and huffman encoding.

Example Operating Modes

- Custom Instruction Extension
    - Vector Dot Product
    - Vector Softmax
- CNN Operation: Convolution
    - Perform convolution on a matrix and return ReLU of the output matrix
    - (note: code may have to repeat custom instruction for each row of the matrix, if ours ASPs handle 1D vectors rather than 2D matrices)
- CNN Operation: Pooling
    - Perform max/avg pooling on a matrix and return the output matrix
    - (note: code may have to repeat custom instruction for each row of the matrix, if ours ASPs handle 1D vectors rather than 2D matrices)
- DSP Operation: DFT
    - Perform DFT on a vector of signal samples and return the magnitudes for the configured frequencies.
- (Optional) DSP Operation: Spectrograph
    - Performed in software:
    - Concatenate DFT outputs where the provided signal is changed between each execution, each an overlapping window of the signal of interest (with window weights applied prior), to create a spectrogram

Appendix

ADMA-MIN Protocol (Nios II, ReCOP, DP-ASP-A/B/C/D, ADC/DAC-ASP)																														
ID Bits								Reserved Data Bits																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	0	1	1	Unuse		WORD		Data Bits																								Data: WORD (length)
1	100 - 111			Reserved				Target Addr				EN	Reserved Config Info																			ASP Config
1	0	0	1	Source Addr				Target Addr				EN	CH	SR	Unused																	Config ADC
1	0	1	0	Source Addr				Reserved				EN	CH	SR	Unused																	Config DAC
Avalon Protocol (with ReCOP)																																Custom Instruction IC

Word Length (WORD)	
0	0
0	1
1	1

Fig. 2. Data package sent over the TDMA-MIN NoC and fields

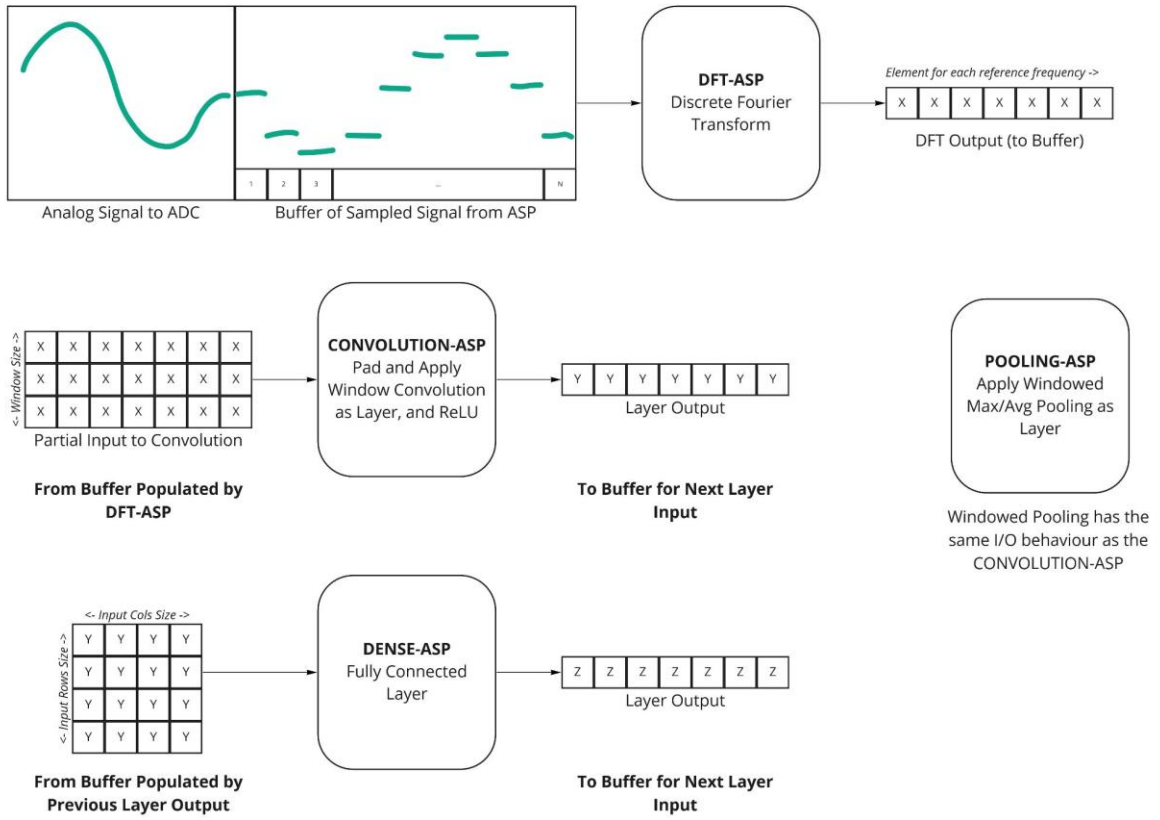

Fig. 3. I/O requirements of ASPs and required buffer registers between ASPs.

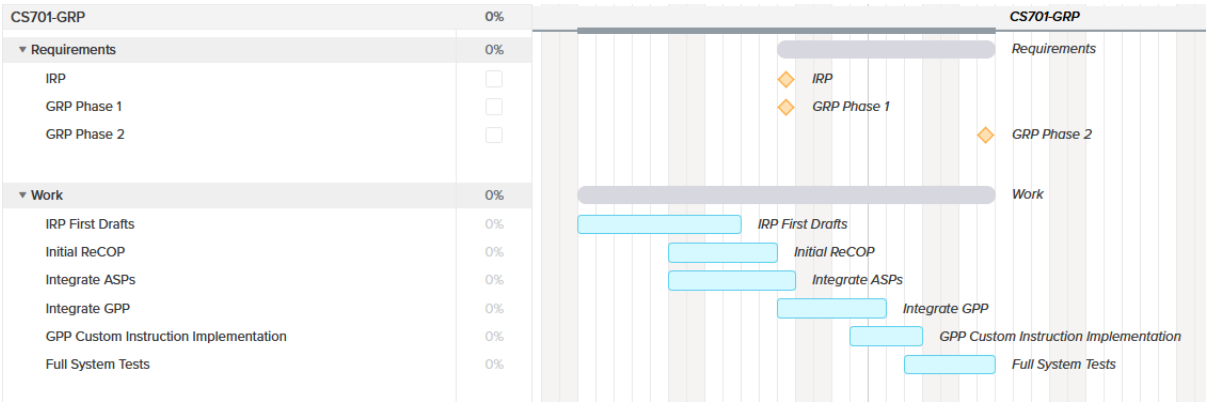

Fig. Gantt Chart of Plan

References

[1] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masera, and M. Martina, 'An Updated Survey of Efficient Hardware Architectures for Accelerating Deep Convolutional Neural Networks', Future Internet, vol. 12, no. 7, Art. no. 7, Jul. 2020, doi: 10.3390/fi12070113.

[2] Z. Wang, X. Li, and J. Zhou, 'Small-footprint Keyword Spotting Using Deep Neural Network and Connectionist Temporal Classifier', arXiv:1709.03665 [cs], Sep. 2017, Accessed: May 09, 2022. [Online]. Available: http://arxiv.org/abs/1709.03665

[3] S. Jagannathan, M. Mody, and M. Mathew, 'Optimizing convolutional neural network on DSP', in 2016 IEEE International Conference on Consumer Electronics (ICCE), Jan. 2016, pp. 371–372. doi: 10.1109/ICCE.2016.7430652.

[4] CS231n Convolutional Neural Networks for Visual Recognition'. https://cs231n.github.io/convolutional-networks/ (accessed May 06, 2022).

[5] N. Akhtar and U. Ragavendran, 'Interpretation of intelligence in CNN-pooling processes: a methodological survey', Neural Comput & Applic, vol. 32, no. 3, pp. 879–898, Feb. 2020, doi: 10.1007/s00521-019-04296-5.

[6] Y.-L. Boureau, J. Ponce, and Y. LeCun, 'A Theoretical Analysis of Feature Pooling in Visual Recognition', p. 8.

[7] H. Park, D. Kim, and S. Kim, 'Digital Neuron: A Hardware Inference Accelerator for Convolutional Deep Neural Networks', ArXiv181207517 Cs Eess, Feb. 2019, Accessed: May 06, 2022. [Online]. Available: http://arxiv.org/abs/1812.07517

[8] Krizhevsky et al., 'ImageNet classification with deep convolutional neural networks | Communications of the ACM'. https://dl.acm.org/doi/10.1145/3065386 (accessed Mar. 21, 2022).

[9] G. Zhao, Z. Zhang, H. Guan, P. Tang, and J. Wang, Rethinking ReLU to Train Better CNNs. 2018, p. 608. doi: 10.1109/ICPR.2018.8545612.