PÁZMÁNY PÉTER CATHOLIC UNIVERSITY
ROSKA TAMÁS DOCTORAL SCHOOL OF
SCIENCES AND TECHNOLOGY

FÜLÖP András

Optimization of Convolutional Neural Networks

PhD Dissertation

Thesis Supervisor:
HORVÁTH András, Ph.D.

Co-supervisor:
CSABA György, Ph.D.

2024

Introduction

Nowadays, we can find acceptable algorithmic solutions for countless complex problems, including NP-complete problems. With the help of deep learning methods, we were able to reach or exceed human performance in lots of areas.

Deep learning solutions usually require a large amount of data and impressive computing capacity, which can be done on a CPU or, more typically, on GPUs. Therefore, training deep learning models requires a lot of energy and its carbon footprint is becoming more and more significant.

Take for example the GPT-3 (Generative Pre-trained Transformer 3) created by OpenAI, an autoregressive language model with 175 billion parameters which achieves strong performance on various NLP problems. Its estimated energy consumption due to training is 1287 MWh and carbon emissions are 552 tCO_2.

Therefore, we can conclude that even in the case of the deep learning solutions used successfully today, energy consumption is a very important aspect and in some cases (e.g. in the case of embedded systems or smartphones) it is a fundamentally decisive factor.

Theses of my dissertation

First Thesis

I implemented a convolutional neural network in the frequency domain without using any inverse Fourier transformation, including the classification part as well.

I have introduced an alternative realization of the spatial activation functions in the frequency domain, and I have presented a possible solution to eliminate the inverse Fourier transformation before the fully connected classification layer.

My neural network architecture was tested on one- and two-dimensional datasets and compared with similar network implementations containing inverse Fourier transformation. The proposed framework could achieve similar or better accuracy without the computational cost of inverse Fourier transformation. In the case of the MNIST dataset, the maximum accuracy of the architecture with inverse FFT decreased by about 6% from the time domain reference (where the maximum was

98.75%), while the maximum accuracy of my solution dropped by just approximately 4%.

In terms of computational efficiency, my model significantly reduced the number of multiplications required. For an input of size 28x28 with 3x3 kernels, the number of multiplications in the Fourier domain was 3136, compared to 7056 in the time domain. This reduction in computational cost, coupled with the comparable or superior accuracy of my model, demonstrates the effectiveness of my approach.

Methods in the frequency domain

My method is based on the convolution theorem, which states the following:

$$\mathcal{F}\{f \star g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}, \tag{1}$$

where $\mathcal{F}$ denotes the Fourier transforms of the f and g functions, $\star$ is the convolution and $\cdot$ means the pointwise multiplication operators.

The first and main part of a convolutional neural network is the convolution itself in which we multiply element-wise the images (or time series) by the appropriate values of the convolutional kernels, which were transformed into the frequency domain before the multiplication as well. If we use smaller kernels than the size of the images or the length of the time series, before the transformation, the kernels had to be padded with zeros for the point-wise multiplication. Due to this padding, after the transformation I always perform the multiplication operations with matrices of the same size, thus I can save even more operations if the kernel size is larger. However, since all my network works in the Fourier domain, I applied another technique and generated the kernels directly in the frequency domain instead of transforming them from time-domain using Fourier transform, thus I can save the cost of kernels' transformation during training. In this case, the kernel size is the same as the size of the input. I used this approach in our experiments, as I present in the results section. This step has no effect on inference time, which is one of the most important factors in neural network training, but can reduce training time.

In spectral representation, we can encounter various activation function implementations with the aim of operating similarly to nonlinearities of the time-domain and achieving a similar result in terms of accuracy.

During the Fourier transform, I transfer the original input signal from the set of real numbers to the complex plane, thus the domain of my activation function will also be the set of complex numbers.

My non-linear function called FReLU $f : \mathbb{C} \to \mathbb{C}$ is a nonlinear function, which can be written as follows:

$$f(z) = \begin{cases} z & if \ \ |z| > \alpha \\ \mathbf{0} & otherwise \end{cases} \tag{2}$$

where $z \in \mathbb{C}$ is equal to $a + ib$ complex number, the $|z| = \sqrt{a^2 + b^2}$, $\mathbf{0}$ is the $(0, 0)$ point in the complex plane and α is a tuneable parameter of this method. This solution can be considered as a high-pass filter with the α cut-off point or as an equivalent of the traditional ReLU funciton for complex numbers. The Fig. 1 illustrates how this function maps the complex plane in case of $\alpha = 0.1$. During our training on the different datasets, the α parameter was 0.1.

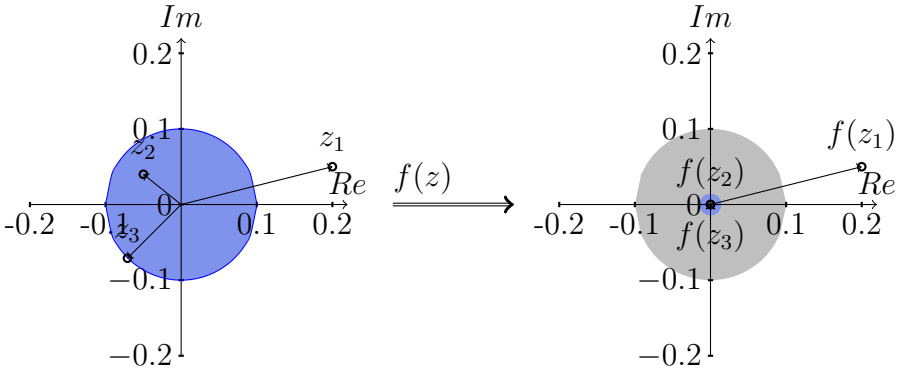

Figure 1: The nonlinear activation function f (with $\alpha = 0.1$) maps z_1 to z_1 and z_2, z_3 to zero, where $|z_1| > 0.1$, $|z_2| < 0.1$ and $|z_3| = 0.1$. The position of points outside the blue circle does not change, but all points in the circle will be zero.

I used the spectral pooling method as a subsampling procedure. In this case, the dimensionality reduction is in the Fourier domain, where the $N \times M$ matrix input is truncated and only the central $H \times W$ submatrix of frequencies remains.

Before I flatten the feature map of the last convolution layer, I calculate the magnitude of the complex values applying a $f_{abs^2} : \mathbb{C} \to \mathbb{R}$ function, which can be written as follows:

$$f_{abs^2}(a + ib) = a^2 + b^2 \tag{3}$$

This is similar to the previously introduced activation function, but the output is the square of the absolute value, which is a real number. The computational complexity of this calculation is $\mathcal{O}(n)$ instead of inverse FFT's $\mathcal{O}(nlog(n))$. After the flattening step I used a traditional fully connected neural network with only one layer to predict the classes.

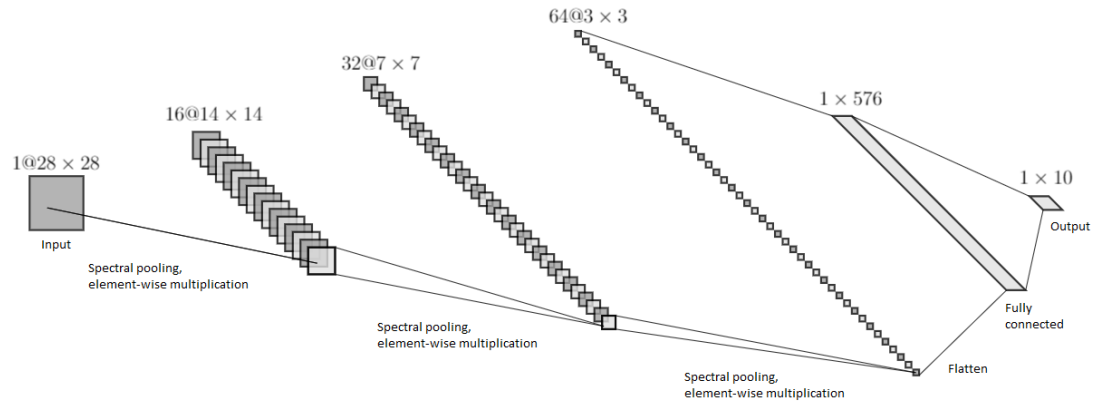

Figure 2: The schema of the proposed CNN architecture. The input is in the frequency domain and the spectral pooling can be done before the element-wise multiplication, and the nonlinear activation function can be applied after each multiplication.

Results

I examine how my architecture works on one- and two-dimensional datasets. For demonstration, in the time domain, I also implemented a CNN, that has the same computational complexity as my neural network in the frequency domain(I used max pooling and ReLU in the time domain), the accuracy results obtained by these CNNs on various datasets can be found in the Table 1.

Table 1: The table contains the average and maximum (I made five different trainings) of the accuracy achieved on the independent test sets of the examined datasets in case of 3 different network architectures. One is the reference network in time domain, the other contains the inverse FFT, which was also used in previous articles, and the neural network implemented with the sum of squares solution proposed by me.

	inverse FFT		sum of squares		time domain	
Dataset	mean	max	mean	max	mean	max
MNIST	90.20%	92.39%	91.93%	94.99%	97.17%	**98.75%**
Fashion-MNIST	80.31%	81.95%	75.34%	82.83%	94.55%	**95.54%**
HADB	92.33%	94.08%	90.54%	93.95%	94.6%	**95.95%**
OZONE	90.26%	96.4%	96.07%	96.4%	94.31%	**97%**

In every case, I made five different trainings and I determined the maxima, the minima, and the average values of these. After that, I compared the results of inverse FFT version with the method I proposed and in the MNIST, Fashion-MNIST and OZONE cases I found (see the Table 1) that the maximum value was higher (or same in case of the maximum of OZONE) in the case I proposed than the inverse FFT method, and only the accuracy of HADB was worse. However, the number of calculations decreased in each case, as instead of $\mathcal{O}(nlog(n))$, only $\mathcal{O}(n)$ operation had to be performed after the convolutional layers (, where n is the size of a sample).

Although the neural network in the time domain outperformed the accuracies of the two frequency-based implementations, in this case much more multiplication is required, as in time domain the computational complexity of convolution is $\mathcal{O}(nm)$, where $m(= H \times W)$ is the size of the kernel, but in the frequency domain, I have only $\mathcal{O}(\frac{n}{4})$ complexity, since, in the frequency domain, the spectral pooling can be executed before the element-wise multiplication. In the frequency domain, the FFT also requires computation ($\mathcal{O}(nlog(n))$), but this can be done (and stored) before the training.

Second Thesis

I introduced a special convolutional neural network with novel kernel convolution, which can be implemented with a wave-based device based on the principles of surface acoustic wave convolvers.

I tested my neural network architecture on one- and two-dimensional datasets, and it was compared with similar network implementations containing normal convolution. The proposed framework could achieve a similar or slightly worse accuracy, but it has the potential to be implemented in a much faster and more energy-efficient device.

When tested on the MNIST dataset, my network achieved a mean accuracy of 86.51% and a maximum accuracy of 93.58%, compared to the reference network's mean accuracy of 92.61% and maximum accuracy of 96.52%. Similar trends were observed with the Fashion-MNIST and HADB datasets, with an average performance drop of approximately 6%.

My results also revealed some of the required properties of future magnetic devices. To ensure high accuracy, the attenuation parameter cannot be lower than $e^{\frac{-i}{999}}$

Methods

I proposed a special convolutional neural network architecture, which is inspired by physical ideas and does not contain additional classical nonlinear activation functions (like ReLU or sigmoid), but the system contains nonlinearity through the physical properties of the simulated device and these characteristics will determine the attenuation and saturation of the convolutional/kervolutional layer.

During the implementation of our neural network, the primary consideration was to examine the physical effects that a device - specifically developed to perform the operation of convolution - may have on an ideal, theoretical artificial neural network.

The real-time SAW convolver, which was the starting point in the implementation of my neural network architecture, can perform convolution only on one-dimensional inputs.

Thus for hardware considerations, I made a one-dimensional convolutional neural network. During my simulations, both one- and two-dimensional datasets were investigated. I have converted the 2D input data and the convolutional kernels to one-dimensional vectors and mapped them onto my simulated devices.

One of the main parts of a CNN is the convolutional layer. The convolution of functions f and g in one dimension can be described as the following:

$$f \star g = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{4}$$

Since my input signal is finite, the value of the function f is zero outside a certain interval (for example $[0, t]$). This way the value of the convolutional integral is also zero in this interval, so the formula can be rewritten as:

$$[f * g](t) = \int_{0}^{t} f(\tau)g(t-\tau)d\tau \tag{5}$$

This operation can be implemented by real-time SAW convolvers, such as three-port elastic SAW convolver (3) under nonlinear operation.

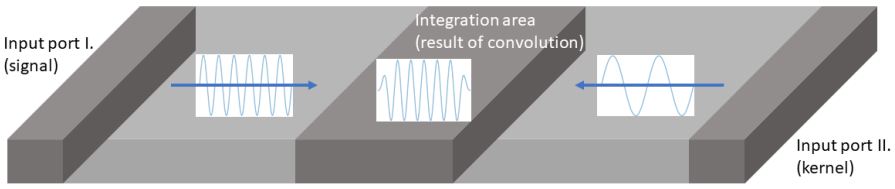

Figure 3: This figure depicts a primitive three-port elastic SAW convolver. Two signals travel in opposite directions form the two inputs ports of the device (at the left and right edges of the device) and the convolved version of the two signals can be extracted at the integration area (in the middle of the device). This example shows how a physical system can be used to implement a complex operation in an energy efficient manner.

The first port of such a device is the input signal port, the second port is the kernel port and between these is the third one, the output or result port. The input and kernel signals can be invoked at the edges of the device by external excitation and the magnetic or electrical changes can be read out from the result port.

Using the Euler formula port I. can be expressed at time t along the z reference axis as:

$$s(t,z) = S(t - z/\nu)e^{j(\omega_0 t - \beta z)} \tag{6}$$

where $S(t - z/\nu)$ the signal modulation envelope is a function of SAW velocity, where $\nu = f\lambda$ and $\beta = 2\pi/\lambda$.

The output of port 2 can be similarly expressed as:

$$r(t,z) = R(t + z/\nu)e^{j(\omega_0 t - \beta z)} \tag{7}$$

where the sign of z is minus, since the signal propagates to the opposite direction.

In this case, the following waveform can be read out from the output port over the length L of the thin-film metal plate:

$$C(t) = P \int_{-\frac{L}{2}}^{+\frac{L}{2}} S(t - z/\nu)R(t + z/\nu)dze^{j2\omega_0 t} \tag{8}$$

where P is a constant, dependent on the nonlinear interaction strength. We can use a change of variable $\tau = (t - z/\nu)$ and reformulate this equation as the following:

$$C(t) = Mve^{j2\omega_0 t} \int_{-\infty}^{+\infty} S(\tau)R(2t - \tau)d\tau \tag{9}$$

where S is the input signal, and R is the kernel signal, M is a constant dependent on the strength of the nonlinear interaction and v is the velocity of the waves (signals), j is the complex unit and ω_0 is the angular frequency of the signal.

The equations (4) and (9) differ only in two factors: the nonlinear dampening ($Mve^{j2\omega_0 t}$) at the beginning of the formula and that the argument of kernel (R) has $2t$ instead of t. The reason for this difference (time compression) is that the

signals are traveling towards one another, (their relative velocity is $2v$), thus the interaction is over in half the time.

In my calculations I have studied a device that works similar to real-time SAW convolvers but the wave exhibits strong damping - therefore the model is well applicable to spin-wave-like convolvers, where damping is more significant.

In my simulation, which can be considered as a baseline, a square signal ($s(t) = A_1 cos(\omega t)$) and a triangular signal ($r(t) = \frac{1}{t} A_2 cos(\omega t)$) travel opposite to each other and the waves propagating in a nonlinear manner. Square and triangular signals were selected as case studies, since they can be easily described mathematically and depict the effect of convolution fairly well. Reading the signal at the intersection of the waves yields the convolution of the two input signals. (In fact, one of the input signals must be inverted in time to obtain convolution, otherwise, we get the cross-correlation of the signals.) The simulation is illustrated in figure 4. The signal is oscillatory, but if we take advantage of the fact that the frequency of the output signal will be twice the original frequency of the signals, we can filter the output signal and we get the convolution result.

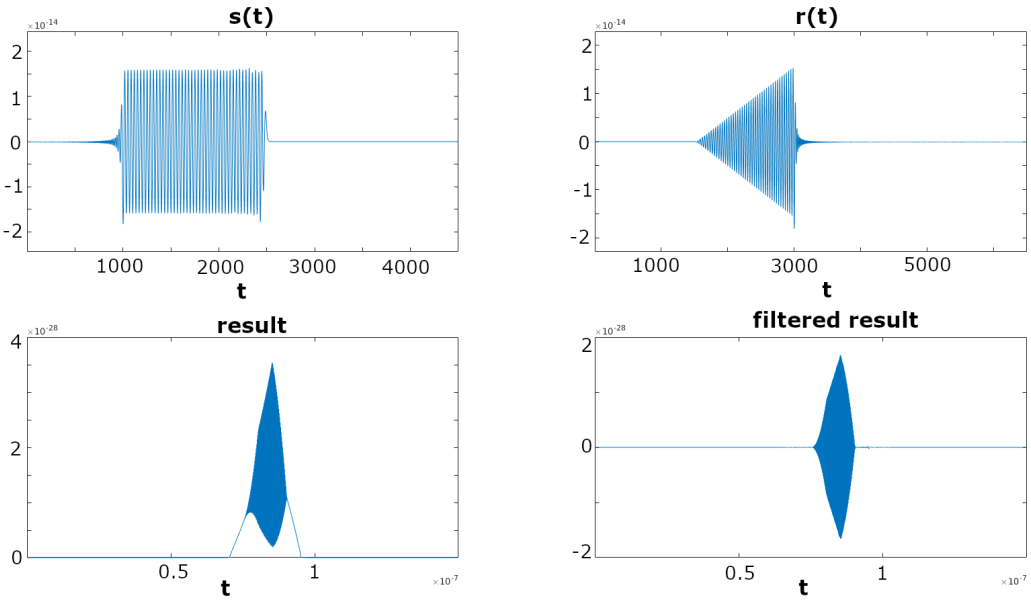

Figure 4: In the first row, $s(t)$ is the square signal, and next to this function is the triangular signal $r(t)$ (inverted in time), these are travelling opposite each other. The first plot in the second row is the raw result, which is read from the collision of the above signals. The last plot is the frequency filtered result, which has doubled frequency compared to the original signals.

In the physical system input signals attenuate over time as they travel further and further in space. Taking this phenomenon into account, we applied exponential attenuation to both the input signal and the kernel.

According to the properties of my physical system, we had to apply saturation after the element-wise multiplication. In fact, I used the following kernel convolution (i_{th} element of convolution) in our CNN architecture:

$$g_i(x) =< \phi_i(x_i), \phi_i(w) > \tag{10}$$

where $< \cdot, \cdot >$ is the inner product of two vectors with hyperbolic tangent (which means $< a, b >= \sum_{k=1}^{n} tanh(a_k b_k)$ and $tanh$ is the saturation of the system), and $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ is the following nonlinear mapping function:

$$\phi_i(x) = e^{\frac{-i}{a}} x_i \tag{11}$$

where i is the discrete time, a is the attenuation parameter. Fig. 5 depicts the $e^{\frac{-i}{a}}$ function with different a parameter. (This attenuation formula can also be written in the following way: $0.999^i x_i$, which means $\phi_i(x)$ with $a = 999$ parameter.)

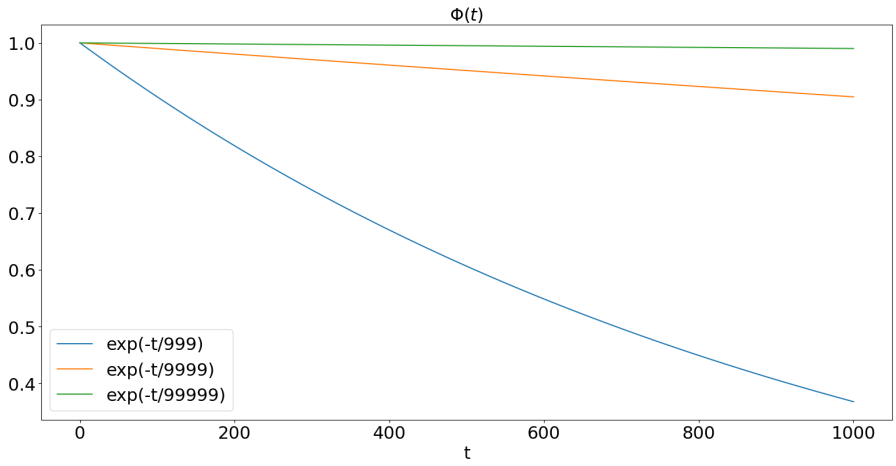

Figure 5: One of the parameters of my simulation is attenuation. The propagating waves were attenuated exponentially with the function $e^{\frac{-i}{a}}$, where i is the time and a is the attenuation parameter. Here we plot the function with 999, 9999 and 99999 attenuation parameters. Since the dependence of attenuation is exponential on this parameter it can heavily affect the accuracy of a neural network.

Results

I started from a simple convolutional neural network and our goal was to implement an architecture which includes a special convolutional operation that could be accomplished by a physical device, which can effectively perform the convolution, thus I introduced physical characteristics into the system to demonstrate the effects of these features. Then I examined how my architecture (depicted in the Fig. 7) works on several one- and two-dimensional datasets. For demonstration, I also implemented a CNN, that is similar to my neural network but uses one-dimensional convolution. I used 1×9 kernels and 3 layers (two layers with 8 kernels and one layer with 16 kernels), and after every convolutional layer, I applied ReLU as nonlinear activation function in the reference CNN model, as shown in the Fig. 6. For the detailed parameter settings of both the network architectures and the training algorithms please take a look at the source code of my neural network

model which can be found in the following GitHub repository: `https://github.com/andfulop/SpinWaveConvolver`. The classification accuracy results of these CNNs on various datasets can be found in Table 2.

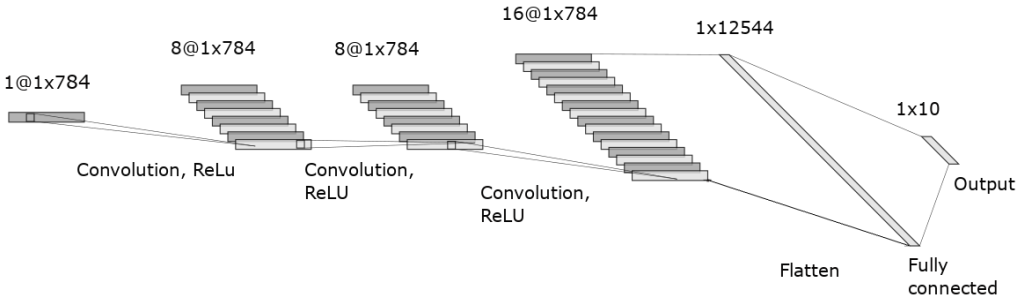

Figure 6: The architecture of my reference neural network model. My network contain three convolutional layers with ReLus followed by a fully connected layer. This simple four layered architecture is capable of solving simple classification tasks.

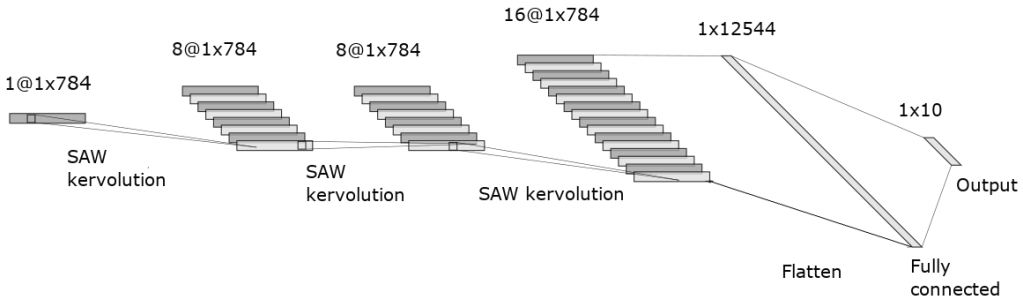

Figure 7: The architecture of my neural network model with SAW kervolution. The convolutions and ReLUs are substituted with kervolutions in this variants. Please note that the number of layers, channels and parameters are the same in both network variants.

	Reference network		metwork with SAW kervolution	
Dataset	mean	max	mean	max
MNIST	92.61%	96.52%	86.51%	93.58%
Fashion-MNIST	77.84%	83.01%	72.87%	79.32%
HADB	88.43%	91.71%	82.11%	88.89%
OZONE	99.15%	99.2%	99.07%	99.4%

Table 2: This table displays the test accuracies of a traditional convolutional network (as the reference) and my method using an SAW convolver in the different columns. The rows contain the accuracies on four different datasets. As it can be seen from the results the same network provided different mean accuracies on different problems ranging from 77 to 92% depending on the complexity of the exact task. One can observe an approximately 6% performance drop in almost all cases (except the OZONE dataset) and this drop is independent from the original accuracy of the reference network.

The earlier results demonstrate that one can substitute convolution with kervolution for a 6% drop in accuracy, which could enable the energy efficient implementation of simple neural networks with SAW convolver. Unfortunately in an ideal neural network signals propagate with infinite speed and without attenuation and noise. To demonstrate the practical usability of an SAW I have investigated how an SAW with different attenuation parameters would perform on the MNIST and HADB datasets. As these plots demonstrate if the attenuation parameter (a) is larger than 9999 the network reaches similar accuracy as reported in Table 2 and a decrease from 99999 to 9999 does not have any affect to the classification accuracy of the network. In case of the further decrease, as in case of $a = 999$ the accuracy of my implementation drops significantly. This can help in the physical design of the SAW convolver and one can select materials and frequencies, where this small level of attenuation is ensured.

Conclusion

The dissertation introduces two novel approaches to optimize convolutional neural networks (CNNs).

The first approach focuses on implementing the entire training process in the frequency domain without using any inverse Fourier transformation. By intro-

ducing an alternative realization of spatial activation functions in the frequency domain, the dissertation achieves similar accuracy without the computational cost of inverse Fourier transformation. The proposed framework is tested on one- and two-dimensional datasets, and the results demonstrate its effectiveness.

The second approach introduces a special CNN architecture with a wave-based convolver, inspired by physical ideas. Instead of using classical nonlinear activation functions like ReLU or sigmoid, the system incorporates nonlinearity through the physical properties of the simulated device. This allows the implementation of a convolution operation that can be performed by a physical device, potentially leading to more energy-efficient implementations on embedded devices. The accuracy of this approach is slightly lower compared to traditional CNNs. However, it opens up possibilities for low-energy implementations.

Publications

Journal Papers

1. Fülöp, A., Csaba, G., and Horváth, A., "A Convolutional Neural Network with a Wave-Based Convolver." Electronics, 12(5), 1126, 2023.
2. Fülöp, A., Horváth, A., "End-to-end Training of Deep Neural Networks in the Fourier Domain.", MDPI Mathematics, 2022.

Conference Papers

1. Fülöp, A., Horváth, A., "Application of Cellular Neural Networks in Semantic Segmentation.", IEEE International Symposium on Circuits and Systems, 2021.
2. Fülöp, A., Horváth, A., "Template Optimization in Cellular Neural Networks Using Gradient Based Approaches." 2020 European Conference on Circuit Theory and Design (ECCTD). IEEE, 2020.