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

# Contents

# Chapter 1

# 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 areas such as chess [1] or Go [2].

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 [3] and its carbon footprint is becoming more and more significant [4].

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, like translation, question-answering, cloze tasks, unscrambling words, using a novel word in a sentence and performing 3-digit arithmetic. [5] Its estimated energy consumption due to training is 1287 MWh and carbon emissions are 552 $tCO_2$. [6]

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.

## 1.1 Outline of the thesis

This dissertation is organized as follows: Chapter 1 presents the thesis points of the dissertation, with the first point based on the findings from [7] article, and the second point based on the findings from [8] article. This chapter also provides an introduction to the field of Neural Networks, tracing its history and exploring its various fields of application.

Chapter 2, based on the article "End-to-End Training of Deep Neural Networks in the Fourier Domain" by Fülöp and Horváth (2022), explores the optimization of Convolutional Neural Networks (CNNs) in the Fourier Domain. It begins with an introduction to the topic, followed by a detailed discussion on the acceleration of networks in the Fourier Domain. The chapter then presents the methods used, including the Convolution Theorem and methods in the frequency domain. The results and discussions section provides insights into the application of these methods on one-dimensional and two-dimensional datasets. The chapter concludes with a discussion and conclusion of the findings.

Chapter 3, based on the article "A Convolutional Neural Network with a Wave-Based Convolver" by Fülöp, Csaba, and Horváth (2023), introduces a Convolutional Neural Network with a Wave-based Convolver. The chapter begins with an introduction, followed by a detailed explanation of the methods used. It presents a one-dimensional network and discusses the convolution and SAW-based kervolution. The chapter then presents the results of the study, including the application of the methods on various datasets. The chapter concludes with a summary of the findings.

In Chapter 4, using the combination of machine learning and temporal DMD amplitude changes, I introduce a method. In this method, I focused on classifying audio files, specifically recognizing the difference between the vowels 'ae' and 'ah', using the architecture I developed. My experiments showed that this architecture can be effective for certain signal classification tasks. This chapter was created at the end of my PhD studies as part of a separate project. While it is partly related to the previous chapters, it can be considered as an independent project, from which no thesis points were ultimately derived.

## 1.2 Thesis points of my dissertation

### 1.2.1 First point

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.

### 1.2.2 Second point

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 $e^{\frac{-i}{999}}$.

## 1.3 Brief overview of Neural Networks

### 1.3.1 History of Deep Learning

The researchers and philosophers have been trying to understand for a long time how human brain and neural system function and they end up in two different tracks: connectionism and computationalism. In case of the connectionism the human intellectual abilities can be explained with a network of very simple units (these units are the neurons). [9]

Alexander Bain (1818-1903) related the processes of associative memory to the distribution of activity of neural groupings and proposed a mode of storage capable of assembling what was required. He described the computational flexibility that allows a neural grouping to function when multiple associations are to be stored. [10]

Warren McCulloch and Walter Pitts speculated on the inner workings of neurons and introduced a primitive neural network model in 1943, which could be described as:

$$y = \begin{cases} 1, & \text{if } \Theta \leq \sum_i w_i x_i \text{ and } z_j = 0 \text{ for all } j \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

where y is the output, $x_i$ is the input of signals, $w_i$ is the corresponding weights, $z_j$ is the inhibitory input and $\Theta$ is the threshold. [11]

Donald O. Hebb (1904-1985) the father of Neural Networks introduced the Hebbian Learning Rule in 1949. Hebb stated that: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that As efficiency, as one of the cells firing B, is increased." So the connection between two units should be strengthened as the frequency of co-occurrences of these two units increase. [10]

Frank Rosenblatt introduced the perceptrons in 1958. He constructed an electronic device named Perceptron that showed the ability to learn according to associations. The Perceptron has four units, first is the sensory unit, and the second one is the projection unit which receives the information from the sensory unit and passes it onto the association unit, which adds the data with different weights and then passes the results to response units. [12]

The perceptrons side by side form single one-layer neural network and the one-layer neural networks form a multi-layer neural network, which is called multi-layer perceptrons (MLP). MLP has universal approximation property, which means MLP can represent any functions, but the universal approximation properties of shallow neural networks need exponentially many neurons. [10]

The backpropagation algorithm, a cornerstone in the development of neural networks, utilizes the propagation of the network's error backwards for the purpose of weight adjustment. This algorithm neural network specific discussion was initially introduced by Paul Werbos in 1974 [13]. The backpropagation algorithm has since become a fundamental component in the training of deep neural networks.

Hopfield Network (introduced in 1982) is a fully connected neural network where the weights connecting the neurons are bidirectional, and the values of units are either 0 or 1. Hopfield Network has content-addressable memory property. [14]

LeNet, which was invented by Le Cun et al., was the first convolutional neural network and it was inspired by the Neocogitron. The architecture of LeNet consists of two convolutional layers and after the convolutional layers, it has downsampling layers as well. The end of this model contains a fully connected layer and an RBF layer for classification. [15]

Reinforcement Learning (RL) is a branch of machine learning that focuses on decision-making processes, where an agent learns to make decisions by interacting with its environment and receiving feedback in the form of rewards or penalties. This learning paradigm has been successfully applied in various domains, including game-playing, robotics, resource management, and recommendation systems. The introduction of RL marked a significant shift in the field of machine learning, moving away from supervised learning paradigms, where an agent learns from a set of labeled examples, towards a more interactive and dynamic learning process. This shift has allowed for the development of more complex and adaptive systems, capable of learning and improving their performance over time through interaction with their environment. [16] The application of RL extends beyond single-agent scenarios. In the survey "Multi-agent deep reinforcement learning: a survey" by Sven Gronauer and K. Diepold, the authors provide an overview of the current developments in multi-agent deep reinforcement learning, where multiple agents learn to interact with each other and the environment. They discuss the unique challenges that arise in the multi-agent domain and review methods to cope with these challenges [17]. RL has also been applied in the realm of autonomous vehicles and game-playing. In the paper "Outracing champion Gran Turismo drivers with deep reinforcement learning" by Peter R. Wurman et al., the authors describe how they trained agents that can compete with the world's best e-sports drivers in the PlayStation game Gran Turismo. They demonstrate the capabilities of their agent, Gran Turismo Sophy, by winning a head-to-head competition against four of the world's best Gran Turismo drivers [18]. In the field of Unmanned Aerial Vehicles (UAVs), RL has been used to ensure a substantial level of autonomy. In the paper "Drone Deep Reinforcement Learning: A Review" by A. Azar et al., the authors provide a detailed description of deep reinforcement learning techniques applied to the guidance, navigation, and control of UAVs. [19] These studies underscore the versatility and potential of reinforcement learning in various domains.

Autoencoders, a type of artificial neural network, have emerged as a powerful tool for learning efficient codings of input data, particularly in the realm of dimensionality reduction and feature learning. These networks are designed to encode input data into a compressed representation, and then decode this representation back into the original data format, thereby learning a form of identity function. This process forces the network to capture the most salient features of the input data in the compressed representation, which can be used for tasks such as noise reduction, anomaly detection, and more [20]. The power of autoencoders has been demonstrated in various fields. For instance, Wehmeyer and

Noé applied a modification of an autoencoder to the task of dimension reduction of molecular dynamics data, showing that their time-lagged autoencoder could find low-dimensional embeddings for high-dimensional feature spaces, capturing the slow dynamics of the underlying stochastic processes [21]. In another study, Kiarashinejad et al. used autoencoders to considerably reduce the dimensionality and computational complexity of electromagnetic nanostructure design problems, transforming the conventional many-to-one design problem into a simpler one-to-one problem plus a much simpler many-to-one problem [22]. These studies underscore the versatility and efficacy of autoencoders in tackling high-dimensional data, providing efficient and meaningful representations that can be used for a range of subsequent tasks and application of autoencoders will continue to play a significant role in this field.

The introduction of Recurrent Neural Networks (RNNs) marked a significant milestone in the field of deep learning. Unlike traditional feed-forward neural networks, RNNs possess a unique architecture that allows them to maintain a form of memory by using their output as part of the input for the next step. This characteristic makes them particularly suited for tasks involving sequential data, such as time-series analysis, natural language processing, and speech recognition. However, the practical application of RNNs was limited due to the vanishing gradient problem, which was later addressed with the introduction of Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber in 1997. [23]

Graph Neural Networks (GNNs) have emerged as a powerful tool for learning from graph-structured data, providing a means to encode both node features and topological information. GNNs have been applied across a wide range of domains, from social network analysis to computational chemistry, and have shown significant promise in handling complex relational data. [24] The foundational work of Scarselli et al. ([25]) introduced the concept of GNNs, which aimed to extend traditional neural networks to directly process graphs, thereby capturing the rich relational structure often present in real-world data. This opened up a new paradigm in machine learning, enabling the development of models that could learn from data structured as graphs, going beyond the limitations of traditional data types like vectors or sequences. An application of Graph Neural Networks can be found in the field of recommender systems. With the rapid growth of online information, recommender systems have become indispensable tools for many businesses, such as e-commerce websites and music services. These systems aim to accurately model users' preferences from their historical interactions and static features, and further recommend items that users might be interested in. However, the main challenge in recommender systems is to learn effective user/item representations from their interactions and side information. Recently, GNN techniques have been widely utilized in recommender systems since most of the information in recommender systems essentially has a graph structure and GNNs have superiority in graph representation learning. For instance, the paper "Graph Neural Networks in Recommender Systems: A Survey" by Shiwen Wu et al., provides a comprehensive review of recent research efforts on GNN-based recommender systems. They provide a taxonomy of GNN-based recommendation models according to the types of information used and recommendation tasks. Moreover, they systematically analyze the challenges of applying GNN on different types of data and discuss how existing works in this field address these challenges. They also state new perspectives pertaining to the development of this field [26].

In 2012, Hinton et al. introduced a novel regularization technique known as Dropout, which has since played a crucial role in the progression of deep learning. The Dropout method addresses the problem of overfitting in neural networks by randomly nullifying a certain proportion of input units at each training step. This process effectively results in a unique network configuration at each iteration. The inherent randomness of Dropout prevents intricate co-adaptations among the network's units, which in turn reduces the likelihood of overfitting and enhances the network's ability to generalize. [27]

While the LeNet has been previously mentioned, the evolution of CNNs includes other significant milestones such as the introduction of AlexNet. AlexNet, introduced by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012, substantially improved image recognition performance. This deep learning model revolutionized the field of computer vision. Its architecture, deeper and wider than previous models, utilized innovative techniques such as ReLU activation function and dropout for regularization. The success of AlexNet sparked a wave of interest in deep learning, particularly for applications in image classification and object detection, setting the stage for subsequent developments in CNN architectures. [28]

Another milestone, the introduction of GANs by Ian Goodfellow and his colleagues in 2014 offered a novel approach for generative models, and they are widely used in the generation and modification of images. GANs consist of two neural networks, a generator and a discriminator, that are trained simultaneously. The generator network generates new data instances, while the discriminator evaluates them for authenticity. This adversarial process leads to the generator network creating increasingly better data. GANs have been particularly impactful in the field of computer vision, where they have been used to generate realistic images, perform image-to-image translation, and even generate 3D object projections. [29]

In 2015, Ioffe and Szegedy introduced a method known as Batch Normalization, which has since become a cornerstone in the field of deep learning. This technique, designed to make artificial neural networks more stable and faster, normalizes the inputs of the layers by re-centering and re-scaling them. Batch Normalization operates by using the statistics of a batch of images to normalize the features of an input image, which can introduce noise to the gradient of the training loss, which is important for the optimization and generalization of deep neural networks. [30]

Capsule Networks (CapsNets), a novel form of artificial neural network, were introduced by Geoffrey Hinton and his team in 2017. CapsNets were designed to address some of the limitations of Convolutional Neural Networks (CNNs), particularly their inability to preserve the hierarchical relationships of objects within an image [31]. A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part. The length of the activity vector is used to represent the probability that the entity exists and its orientation to represent the instantiation parameters. Active capsules at one level make predictions, via transformation matrices, for the instantiation parameters of higher-level capsules. When multiple predictions agree, a higher level capsule becomes active. The authors introduced an iterative routing-by-agreement mechanism: A lower-level capsule prefers to send its output to higher level capsules whose activity vectors have a big scalar product with the prediction coming from the lower-level capsule. This mechanism is more effective than the max-pooling in CNNs, which allows neurons in one layer to ignore all but the most active feature detector in a local pool in the layer below. The authors demonstrated that their dynamic routing mechanism is an effective way of segmenting highly overlapping objects. They showed that a discriminatively trained, multi-layer capsule system achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits. In the realm of medical imaging, CapsNets have shown promise. For instance, in the study "Brain Tumor Type Classification via Capsule Networks" by Parnian Afshar, Arash Mohammadi, and K. Plataniotis, CapsNets were used for the classification of brain tumor types. The authors found that CapsNets could successfully outperform CNNs for this classification problem, demonstrating the potential of CapsNets in medical applications [32].

The Transformer models, introduced by Vaswani and colleagues in 2017, which utilize attention mechanisms, have represented a significant advancement in the field of Natural Language Processing (NLP). The Transformer model, unlike its predecessors, does not rely on recurrence but instead uses a self-attention mechanism, allowing it to consider different words in the sentence simultaneously, thereby capturing the context more effectively. This has led to substantial improvements in tasks such as machine translation, text summarization, and sentiment analysis. Furthermore, the Transformer model forms the backbone of more recent models like BERT, which have set new performance benchmarks on a wide array of NLP tasks. [33] [34] [35]

Transfer learning is a powerful technique in machine learning, where a pre-trained model is repurposed for a new problem. This approach has gained significant traction in deep learning, particularly in the fields of computer vision and natural language processing, where pre-trained models serve as a starting point for specific tasks. Transfer learning is particularly useful in scenarios where the training and future data may not be in the same feature space or have the same distribution. In such cases, knowledge transfer can significantly improve the performance of learning by avoiding the need for extensive data labeling efforts. There are several real-world examples where transfer learning can be beneficial. For instance, in the field of web document classification, the distribution of review data among different types of products can be very different, making it expensive to collect a large amount of labeled data to train the review-classification models for each product. In such cases, transfer learning can save a significant amount of labeling effort.[36] In the paper "Learning to Prompt for Vision-Language Models" by Zhou, Yang, Loy, and Liu, the authors explored the use of large pre-trained vision-language models like CLIP, which have shown great potential in learning representations that are transferable across a wide range of downstream tasks [37]. The authors proposed Context Optimization (CoOp), a simple approach specifically for adapting CLIP-like vision-language models for downstream image recognition. Through extensive experiments, they demonstrated that CoOp requires as few as one or two shots to beat hand-crafted prompts with a decent margin and is able to gain significant improvements over prompt engineering with more shots. In another study, "Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning" by Gordon, Duh, and Andrews, the authors explored weight pruning for BERT and its effects on transfer learning [38]. They found that BERT can be pruned once during pre-training rather than separately for each task without affecting performance. This finding is significant as it suggests that the benefits of transfer learning can be achieved with more compact models, making them more feasible for deployment in resource-constrained scenarios. These studies highlight the versatility and potential of transfer learning in deep learning. By leveraging pre-trained models, researchers and practitioners can effectively bootstrap the learning process for new tasks, saving computational resources and potentially achieving superior performance.

## 1.3.2 Fields of Application

In this section, I strive to offer a wide-ranging, though not all-encompassing, overview of the diverse areas where techniques of deep learning and neural networks have been effectively utilized. The ongoing progress in these fields points towards a future full of promise, teeming with vast potential for additional applications and enhancements.

### 1.3.2.1 Natural Language Processing

In the case of Natural Language Processing (NLP), Neural Networks and Transformer-based models have been instrumental in driving significant advancements. These models have been effectively utilized in a variety of NLP tasks, including text classification, machine translation, and cognitive dialogue systems, among others.

The introduction of the BERT (Bidirectional Encoder Representations from Transformers) model by Devlin et al. in 2018 marked a significant milestone in the field of NLP. BERT is a transformer-based machine learning technique for NLP pre-training. It considers the full context of a word by looking at the words that come before and after it—hence the term 'bidirectional'. This approach has led to state-of-the-art results on a wide array of NLP tasks. The BERT model has been adopted by Google in its search engine, demonstrating its practical effectiveness [34].

Following the success of BERT, Facebook AI introduced RoBERTa (A Robustly Optimized BERT Pretraining Approach). RoBERTa, developed by Liu et al., is a variant of BERT that is claimed to have significantly improved performance over its predecessor. The improvements were achieved by modifying key hyperparameters in BERT, including training the model longer with bigger batches over more data, removing the next-sentence pretraining objective, and training on longer sequences [39].

In the research paper "Revisiting Pre-trained Models for Chinese Natural Language Processing," the authors undertake a comprehensive examination of the effectiveness of pre-trained language models, specifically in the context of Chinese, a non-English language. They acknowledge the remarkable advancements that Bidirectional Encoder Representations from Transformers (BERT) have brought about in various Natural Language Processing (NLP) tasks. However, they also recognize the continuous efforts to enhance the performance of these pre-trained language models through the introduction of subsequent variants. In response to this ongoing development, the authors propose MacBERT, a model that builds upon RoBERTa, offering several improvements, particularly in the masking strategy that adopts the Masked Language Model (MLM) as a correction mechanism. The authors' approach with MacBERT (Mac means "MLM as correction") is both simple and effective, demonstrating the potential for significant advancements in the field. The paper presents extensive experiments conducted on eight different Chinese NLP tasks to reassess the existing pre-trained language models and the proposed MacBERT. The results of these experiments reveal that MacBERT can achieve state-of-the-art performances on many NLP tasks. The authors also delve into several findings that may guide future research in this area. This paper not only contributes to the understanding of pre-trained models for Chinese NLP but also provides a valuable resource for the community by releasing the Chinese pre-trained language model series. [40]

OpenAI's GPT-3 (Generative Pretrained Transformer 3) is another significant development in the field of NLP. GPT-3, introduced by Brown et al., is an autoregressive language model that uses deep learning to produce human-like text. With 175 billion machine learning parameters, GPT-3 is capable of tasks such as translation, question-answering, and even writing poetry. [5].

#### 1.3.2.2 Medical Imaging

In medical imaging, Convolutional Neural Networks (CNNs) have been increasingly utilized to aid in the diagnosis and treatment of various health conditions. The ability of CNNs to effectively process and analyze complex image data has made them particularly useful in this field.

One of the key architectures that have been widely adopted in this domain is the U-Net, a type of CNN that was specifically designed for biomedical image segmentation [41]. The U-Net architecture, introduced by Ronneberger et al., is particularly effective for tasks that require the detection of small, localized features, such as the identification of cells or tumors in medical images. The architecture is characterized by its U-shaped design, which consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. This design allows the network to learn from fewer training images and still yield more precise results [41].

A notable application of CNN in medical imaging is the work of Esteva et al., who used a deep learning model for the detection of skin cancer. Their model was trained on a dataset of nearly 130,000 clinical images and was able to achieve a performance on par with dermatologists, demonstrating the potential of CNNs in aiding medical diagnosis [42].

In the realm of tuberculosis diagnosis, a study by Lakhani and Sundaram [43] demonstrated the use of CNNs for the detection of tuberculosis from chest radiographs. The authors used deep convolutional neural networks (DCNNs) to classify images as having manifestations of pulmonary tuberculosis (TB) or as healthy. The best-performing classifier had an area under the curve (AUC) of 0.99, which was an ensemble of the AlexNet and GoogLeNet DCNNs. There were 13 out of 150 test cases where the DCNNs did not agree on the classification. These cases were then independently evaluated by a cardiothoracic radiologist, who was able to correctly classify all 13 cases. This approach, which combined the DCNNs with the expertise of a radiologist, achieved a sensitivity of 97.3% and a specificity of 100%. [43]

In the case of medical imaging, the work of Zhang, Liu, and Shen stands out for its innovative approach to large-scale anatomical landmark detection. In their study, they proposed a two-stage task-oriented deep learning method that can detect a multitude of anatomical landmarks in real-time, even when trained on a limited dataset. The method they developed employs two deep convolutional neural networks (DCNNs), each tailored to a specific task. The first stage involves a CNN-based regression model that learns the inherent relationships between local image patches and the target anatomical landmarks. This is achieved by feeding the model millions of image patches. The second stage introduces another CNN model, which incorporates a fully convolutional network (FCN) that shares the same architecture and network weights as the CNN used in the first stage. This second stage model also includes additional layers designed to predict the coordinates of multiple anatomical landmarks simultaneously. This two-stage approach allows the model to integrate both local and global information into the learning process. Local information, such as the inherent associations between image patches and their displacements to landmarks, is learned in the first stage. Global information, such as the association among image patches, is incorporated in the second stage. The authors demonstrated the effectiveness of their method by applying it to brain landmark detection using MR data, and prostate landmark detection using CT data. Their method achieved impressive accuracy, with a mean error of 2.96 mm and 3.34 mm for brain and prostate landmark detection, respectively. Furthermore, their method was able to detect thousands of landmarks simultaneously in approximately 1 second, showcasing the efficiency of their approach. [44]

In addition to the original U-Net, there have been several variants and improvements to the architecture. One such variant is the UNet++, introduced by Zhou et al. This architecture is a deeply-supervised encoder-decoder network where the encoder and decoder sub-networks are connected through a series of nested, dense skip pathways. The re-designed skip pathways reduce the semantic gap between the feature maps of the encoder and decoder sub-networks. The UNet++ architecture has been evaluated across multiple medical image segmentation tasks, including nodule segmentation in the low-dose CT scans of chest, nuclei segmentation in the microscopy images, liver segmentation in abdominal CT scans, and polyp segmentation in colonoscopy videos. [45]

Another significant application of CNNs in medical imaging is in the field of endoscopy. Based on the study "Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images" by Hirasawa et al., Convolutional Neural Networks have shown significant potential in the field of medical imaging, particularly in the diagnosis of gastric cancer. The researchers developed a CNN that was trained using over 13,000 endoscopic images of gastric cancer. The CNN was then tested on an independent set of stomach images collected from 69 patients with 77 gastric cancer lesions. The CNN was able to correctly diagnose 71 of the 77 gastric cancer lesions, demonstrating a sensitivity of 92.2%. This study illustrates the potential of GNNs in automating the detection of gastric cancer, which could significantly aid screening and evaluation efforts in areas with limited access to radiologists [46].

These examples illustrate the potential of CNNs in the field of medical imaging. As these techniques continue to evolve, they are likely to play an increasingly important role in improving the accuracy and efficiency of medical diagnoses.

#### 1.3.2.3 Autonomous Vehicles and Robotics

In the realm of autonomous vehicles and robotics, Convolutional Neural Networks (CNNs) have emerged as a powerful tool for a variety of tasks, including perception, navigation, and control. A key application of CNNs in this domain is in the processing of visual data to understand the surrounding environment, a task that is crucial for the operation of autonomous vehicles and robots.

Drawing from the insights presented in the paper "End to End Learning for Self-Driving Cars" by Bojarski et al., it is clear that Convolutional Neural Networks (CNNs) can play a pivotal role in the development of autonomous driving systems. The researchers devised a unique approach that allows a CNN to learn how to steer a vehicle by using raw pixels from a single front-facing camera and corresponding steering commands.

Notably, the neural network in this study did not rely on human-labeled data but instead learned directly from raw data, using the human steering angle as the only training signal. The network was able to learn the entire process of driving, from perception to control, without any manual decomposition of the problem.

The architecture of the network they proposed is composed of nine layers, including a normalization layer, five convolutional layers, and three fully connected layers. The 66x200 pixel YUV image, which serves as the input, first passes through the normalization layer and then through the convolutional layers, where the extraction of features occurs. The final steering command is produced by the fully connected layers. To train the network, the researchers collected a diverse set of data by driving under various conditions, including different roads, lighting, and weather. They augmented this data to teach the network how to correct itself from less-than-ideal positions or orientations. The network's objective during training was to minimize the mean squared error between the human driver's steering command and the network's output. The performance of the network was evaluated using a simulator that uses pre-recorded videos from a forward-facing on-board camera on a human-driven vehicle. The simulator generates images that approximate the view if the CNN were controlling the vehicle. It also adjusts the subsequent frame in the test video to reflect the position that the vehicle would be in if it followed the CNN's steering commands. Following the successful simulation tests, the researchers proceeded to conduct real-world road tests. The trained network was loaded onto the DRIVE PX in their test vehicle and evaluated under actual driving conditions. The performance was measured based on the proportion of time during which the car was able to steer autonomously. This excluded instances of lane changes and turns from one road to another. In a typical drive from their office in Holmdel to Atlantic Highlands in Monmouth County, New Jersey, the vehicle was able to operate autonomously approximately 98% of the time. Furthermore, the vehicle was able to drive 10 miles on the Garden State Parkway, a multi-lane divided highway with on and off-ramps, without any manual interventions. This real-world testing further validated the effectiveness of the CNN in controlling the vehicle, demonstrating the potential of this approach for the development of self-driving cars [47].

Another notable example of CNN application in this field is the work of Badrinarayanan et al., who developed SegNet, a deep fully convolutional neural network architecture for semantic pixel-wise segmentation. SegNet was designed to be efficient in terms of memory and computational time during inference, making it suitable for real-time applications such as autonomous driving. The architecture of SegNet includes an encoder network, which is topologically identical to the 13 convolutional layers in the VGG16 network, and a corresponding decoder network. The decoder network uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling, which helps to improve boundary delineation and reduce the number of parameters, enabling end-to-end training. [48]

In the research study "A General Pipeline for 3D Detection of Vehicles" an innovative approach to the 3D perception of vehicles in autonomous driving systems is introduced. The authors identify the accurate detection of vehicles in 3D as a significant challenge in autonomous driving. While 2D vehicle detection methods have been widely implemented, they lack the depth data necessary for autonomous vehicles to perform planning and decision making effectively. To tackle this challenge, the authors propose a flexible pipeline that can incorporate any 2D detection network and fuse it with a 3D point cloud to generate 3D information. This is achieved with minimal modifications to the 2D detection networks. The authors develop an effective model fitting algorithm based on generalized car models and score maps, and propose a two-stage convolutional neural network (CNN) to refine the detected 3D box. The proposed pipeline is tested on the KITTI dataset using two different 2D detection networks. The results demonstrate the flexibility of the proposed pipeline and its effectiveness in 3D detection. The pipeline ranks second among the 3D detection algorithms, indicating its competencies in this field. The paper also provides a comprehensive discussion of the current algorithms for 3D vehicle detection, which can be categorized into four major groups: mono image-based, stereo image-based, LiDAR (Light Detection and Ranging), and fusion between mono image and Lidar. The authors argue that the prior approaches for 3D vehicle detection are not as effective as those for 2D detection, and propose their flexible 3D vehicle detection pipeline as a solution. [49]

The paper titled "Distant Vehicle Detection Using Radar and Vision" presents a novel approach to enhancing vehicle detection in autonomous driving systems. The authors identify a key challenge in autonomous driving, which is the accurate detection of distant vehicles. While image-based object detection methods using convolutional neural networks (CNNs) have shown excellent performance, their effectiveness is reduced when detecting small or distant objects. To address this challenge, the authors propose the integration of radar data with image-based detection methods. Radar data is robust to variable weather conditions and provides accurate measurements independent of range, including direct velocity measurements. This integration significantly improves the detection of distant vehicles, making it a valuable contribution to the field of autonomous driving. The authors also introduce a detector that utilizes both monocular images and radar scans to perform robust vehicle detection across various settings. They present an efficient automated method for generating training data using cameras of different focal lengths. Furthermore, they discuss the creation of their dataset, which they label automatically using an existing detector. This dataset is generated using two cameras configured as a stereo pair and a third camera with a long focal length lens. Additionally, radar data is collected using a Delphi ESR 2.5 pulse Doppler cruise control radar. [50]

Zhao et al. proposed a method called GAN-VEEP (Generative Adversarial Network-based VEhicle trajEctory Prediction) for predicting the future location of vehicles on urban roads. The method consists of three components: vehicle coordinate transformation for data set preparation, a neural network prediction model trained by GAN, and a vehicle turning model to adjust the prediction process. The vehicle coordinate transformation model deals with the complex spatial dependence in the urban road topology, the neural network prediction model learns from the behavior of vehicle drivers, and the vehicle turning model refines the driving path based on the driver's psychology. The experimental results showed that GAN-VEEP exhibits higher effectiveness in terms of average accuracy, mean absolute error, and root-mean-squared error compared to its counterparts. [51]

# Chapter 2

# Convolutional neural network optimization in the Fourier Domain.

## 2.1 Introduction

Neuroscience has inspired artificial intelligence techniques like Convolutional Neural Networks (CNNs), which were motivated by the visual cortex in the brain. CNNs consist of two main alternating parts: the convolutional and pooling layers, like simple and complex cells in the visual cortex [52]. Nowadays CNNs can reach exceptional performance in a wide range of machine learning tasks like image classification and natural language processing. Convolutional layers are still used in most state of the art architectures [53], [54] such as vision transformers [55]. It was also demonstrated in [56] that similar state-of-the-art performance can be reached with highly optimized, purely convolutional architectures. Unfortunately there are limiting aspects of these architectures as well: updating a large number of parameters and executing myriads of multiplications requires significant computational resources.

There are various approaches aiming to decrease the number of operations and by this the inference time of the networks or to reduce their computational need employing architectural changes. For instance, in the case of SqueezeNet [57], Iandola et al. were able to make a small network architecture with AlexNet-level accuracy on ImageNet, by downscaling the number of channels in each layer using 1x1 filters and by this decreasing both the number of operations and trainable parameters simultaneously. Another solution, presented first in MobileNets [58] was described by Howard et al., their architecture contained two hyper-parameters to build small and low latency models for mobile and embedded vision applications.

The goal of Knowledge Distillation [59] was similar, to make a fast and minimized network. Yim et al. introduced a novel knowledge transfer technique, where the transferred distilled knowledge from a pretrained neural network is determined by computing the inner product of features from two layers and by this decreasing neuron or layer numbers.

The invention of Farhadi et al.[60] is based on weight quantization and in a corner case the binarization of the weights and of the intermediate representations of data in a convolutional neural network. This method includes optimization processes to determine the best approximations of the convolution operations in CNN using binary operations.

A novel approach for compressing deep neural networks was introduced in [61], which took into account the nonlinear responses and the multi-linear low-rank constraint in the kernel tensors. They suggest a convex relaxation strategy that can be solved directly using the alternating direction method of multipliers (ADMM) to address the difficulty of nonconvex optimization. As a result, they can determine the feature matrix of the Tucker decomposition [62] and Tucker-2 rank at the same time. The suggested method is tested on ImageNet dataset for CNNs such as AlexNet, ResNet-18 and GoogleNet. This method can achieve a significant decrease in model size while sacrificing just a minor amount of accuracy.

In [63] the researchers minimized the parameters and save operations by modifying the DenseNet deep layer block. This technique can reduce the multiple growths of parameter amount for deeper layers by using channel merging procedures while the accuracy remains relatively unchanged. In the case of DenseNet and RetNet-110, the parameters may be lowered by 30–70%. This lightweight network can be used in real-time on an embedded device.

In [64] the authors presented a novel minimalist hardware architecture called adder convolutional neural network (AdderNet) to reduce the computational complexity and energy burden. In this architecture, they use adder kernel with hardware accelerators instead of original convolution. They can achieve 47.85%-77.9% reduction in power consumption and a 16% increase in speed.

These previous methods are independent from each other and they can be combined with each other, but usually in this case accuracy drops significantly. They all aim the simplification of network structure, merging or completely removing neurons, channels or layers resulting different architectures with lower computational need, but since they simplify the network architecture they are not mathematically equivalent with the original network, but approximate it fairly well. Because of this in most cases they also decrease the accuracy of the networks. In this paper I will demonstrate another method which exploits the fact that convolutions can be implemented as pointwise multiplications (Hadamard products) in the Fourier domain and by this it can also be combined with all previously mentioned approaches and can be generally used to decrease the computational need of a neural network. Unfortunately meanwhile convolutions can be more efficiently executed in the Fourier domain other elements of a typical neural network such as non-linearities and pooling operations can only be executed using significantly more operations in this domain and can be more efficiently applied in the time domain. Most approaches of network optimization, also all methods listed earlier, tries to substitute and approximate convolution in the time domain. My approach follows a different path, where I execute all operations in the Fourier domain where convolution can be efficiently applied and I try to approximate the other operations in the Fourier domain.

There are existing methodologies in the literature which exploit the advantageous property of the Fourier transform or other spectral methods, but all of these substitute only specific computational building blocks in the Fourier domain and return from it with an inverse transformation, which adds extra computation to the system. Some of these go back to the time domain directly after the convolution part to apply the nonlinear activation and the downsampling step (e.g. [65], [66]), but there exist solutions, which provide an approximation to implement pooling and nonlinear activation functions in the frequency domain as well (e.g. [67], [68], [69]), thus even in these architectures one inverse Fourier transformation is applied at the last layer of the network.

The authors [70] investigate the implementation of convolution in the Fourier domain using the FFT transform, but for this, the transformation has to be applied at every kernel.

Similarly, discrete cosine transform (DCT) is used in [71], where the authors suggest a faster convolution method for neural networks. They transform the convolutional kernel and the input into the spectral domain with discrete cosine transform and then perform pointwise multiplication between the feature map and kernel. The complexity of DCT is significantly smaller than the FFT method because discrete cosine transform involves only real arithmetic. They use intrinsically extended kernels to suppress repeated domain transformations and they decrease the kernel symmetry with spectral dropout. This model can accelerate the FFT-based methods without a significant decrease in accuracy.

In [72] the authors proposed a method combining FFT, CNN, and LSTM (long and short-term memory). At first, they convert data to the Fourier domain, then features are obtained by CNN and after that, they complete the fault diagnosis of the circuit with the LSTM network. They improved the quality of CSTV analog circuit fault diagnosis with this FFT-CNN-LSTM method.

All these previously introduced spectral approaches use a spectral transformation and an inverse transformation to return to time domain after convolution, after a layer or after at the end of the network. I will demonstrate in this work that these returns are not necessary and a neural network can be fully trained in the Fourier domain and their weights, which in this case represent the weights of certain Fourier components, can be directly used in the following layer even in the logit layer for classification. . This approach can further decrease the number of required operations and as we will demonstrate it also does not decrease the accuracy of the network significantly.

## 2.2 Acceleration of Networks in the Fourier Domain

The idea, that convolutions can be performed as pointwise multiplications in the Fourier domain in case of a convolutional neural network, appeared before the appearance of large scale benchmark datasets brought the important need of training acceleration, however, the number of feature maps was too small to apply this method effectively. Nowadays the speedup caused by Fourier transformations became significant, Mathieu et al. [65] implemented a Fourier based algorithm which requires $6Cn^2log(n) + 4n^2$ operations instead of the direct method with $(n-k+1)^2k^2$ operations, where my input image has dimensions of the $nxn$, $kxk$ is the size of the convolution kernel and $C$ is the hidden constant in the $\mathcal{O}$ notation.

The main additional cost of the frequency-based method is the Fourier transformation especially in [65] because this solution needs inverse transformation before every non-linear activation part and after that a Fourier transformation again. In [66] two new convolution implementations used together with Fast Fourier transform were introduced and compared. The fbfft outperforms the cuFFT convolution implementation in most deep learning problems (introduced in [65] and [66]), but both of these outperform the original cuDNN variant.

Nevertheless, these transform methods have limitations as well, such as the problem of the number of instructions issued, for example, the throughput for 32-bit floating-point multiply-add operations is greater than the throughput for shuffles [66]. In [73] a technique was presented to mitigate the bottleneck of transformation cost and it was shown, that the "overlap-and-add" technique can reduce the computational time by a factor of up to 16.3 times compared to the traditional convolution implementation in a special case.

However, not only the operation of convolution can be simplified in the frequency domain, but the subsampling process may also change. In [74] the authors used the spectral pooling method instead of the conventional max-pooling as dimensionality reduction. This method is truncating the representation of the data in the frequency domain thereby preserves more information per parameter and enables flexibility in output dimensionality. This spectral representation based pooling method was applied in [75], where the training of FCNN architecture was conducted within the frequency domain without the addition of extra non-linearity.

In [67], the authors introduced a non-linearity in the frequency domain, which was called as Fourier domain exponential linear unit and they used pyramid pooling layers for downsampling in the frequency domain. Ayat et al. (in [68]) introduced the frequency domain equivalent of the conventional batch normalization which increases the accuracy of the network. They used a novel nonlinear activation function, the Spectral Rectified Linear Unit (SReLU) after the Spectral pooling. Following the last convolutional layer but before the fully connected layer and softmax layer they executed an inverse Fast Fourier Transform to get real numbers instead of the complex valued representation. Because of this step this approach used unnecessary computation and can not be considered a fully, end-to-end spectral training.

In [69] an other kind of spectral ReLU operation was proposed (called 2SReLU), that adds low frequency components with their second harmonics, and this method has two hyperparameters to adjust each frequency contribution to the final result. The equation of 2SReLU is as follows: $F(\mu_1) \leftarrow \alpha F(\mu_1) + \beta F(\mu_2)$

All of these approaches presented implementations of convolution, pooling algorithms or non-linear activation functions, but all of them applied an inverse transformation after the steps were executed. In the next section, I propose a convolutional neural network architecture, which is entirely in the Fourier domain and it contains pooling, nonlinear activation function, and batch normalization in the spectral domain, and it calculates the fully connected layer and softmax layer without spectral-spatial domain switching as well. For the sake of reproducibility the source code of my neural network, which contains the exact network architectures used in training and a detailed list of training parameters for my experiments can be found in the following GitHub repository: `https://github.com/andfulop/TrainingInFourierDomain`.

## 2.3 Methods

### 2.3.1 Convolution Theorem

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

$$\mathcal{F}\{f \star g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}, \tag{2.1}$$

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 theorem is also true backward, in the time domain the pointwise multiplication is convolution in the frequency domain.)

### 2.3.2 Methods in the frequency domain

During the simulations, my datasets were traditional two-dimensional grayscale images (matrices) and one-dimensional time series (vectors). Therefore, at first, a discrete one- or two-dimensional Fourier transform was applied to these datasets accordingly. After the transformation, each one of the values was represented by a complex number and all operations of the convolutional neural network were executed in the frequency domain.

#### 2.3.2.1 Convolution operation

The first and main part of a convolutional neural network is the convolution itself in which I 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 I 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 my 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.

#### 2.3.2.2 Nonlinear activation function

A sufficiently large neural network using non-linearity can approximate arbitrarily complex functions ([76], [68]) furthermore the learning dynamics and the expressive power of the network depends heavily on the applied non-linearity. The activation function $\Phi : \mathbb{R} \rightarrow \mathbb{R}$ maps the input of a neuron into a specific range and this value is the output of the cell.

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. One of these solutions is the Fourier domain exponential linear unit (FELU, [67]), which is the spectral equivalent of the exponential linear unit (ELU) of the spatial domain. The ELU can be defined as the following:

$$f(x) = \begin{cases} x & if \;\; x > 0 \\ a(e^x - 1) & otherwise \end{cases} \tag{2.2}$$

Another spectral activation function is the Spectral ReLU (SReLU, [68]). This method uses the following polynomial to approximate the traditional ReLU function: $c_0 + c_1X + c_2X^2$. Of course, considering that the multiplication of two signals in the spatial domain is equivalent to the convolution of two signals in the Fourier domain, thus, the previous equation can be modified as follows: $c_0 + c_1X + c_2(X \star X)$, where the $\star$ denotes the convolution operator.

**Our implementation:**
I took the simplicity of the ReLUs (like $max(0, x)$, or $max(x, ax)$, where $a < 1$) methods as a basis, more precisely, the simple and efficient computation (multiplication, addition, and comparison) of it.

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.

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

$$f(z) = \begin{cases} z & if \;\; |z| > \alpha \\ \mathbf{0} & otherwise \end{cases} \tag{2.3}$$

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 $\alpha$ is a tuneable parameter of this method. This solution can be considered as a high-pass filter with the $\alpha$ cut-off point or as an equivalent of the traditional ReLU funciton for complex numbers. The Fig. 2.1 illustrates how this function maps the complex plane in case of $\alpha = 0.1$. During my training on the different datasets, the $\alpha$ parameter was 0.1.

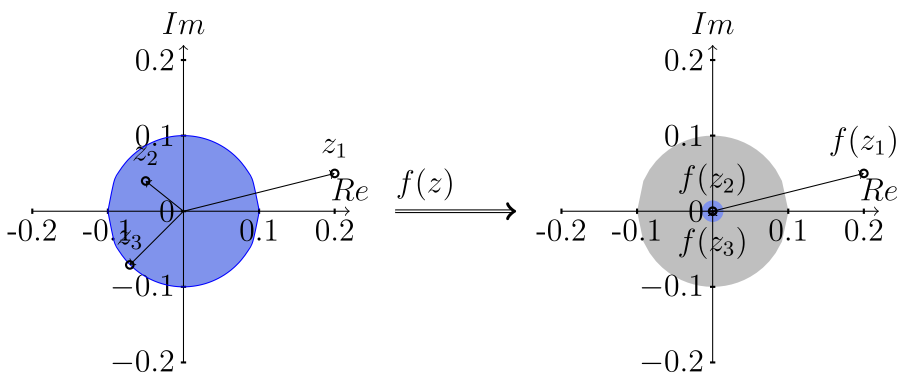


Figure 2.1: The nonlinear activation function f (with $\alpha = 0.1$) maps $z_1$ to $z_1$ and $z_2$, $z_3$ to zero, where $|z_1| > 0.1$, $|z_2| < 0.1$ and $|z_3| = 0.1$. The position of points outside the blue circle does not change, but all points in the circle will be zero.

#### 2.3.2.3 Subsampling operation

I used the spectral pooling method introduced in [74] 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. This approach is different from other pooling strategies in the time domain, such as max pooling, which reduces dimensionality at least a factor of 4 in two-dimensional cases, and the maximum value in each window sometimes does not represent well enough the contents of the window. In contrast, spectral pooling can tune the output dimensionality, and besides, it can be considered as a filter as well, because the removed higher frequencies encode noise in the two-dimensional case [74].

#### 2.3.2.4 Classifier

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

$$f_{abs^2}(a + ib) = a^2 + b^2 \tag{2.4}$$

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 $\mathcal{O}(nlog(n))$. (Previous solutions introduced by others used an inverse Fast Fourier Transform.) After the flattening step I used a traditional fully connected neural network with only one layer to predict the classes.

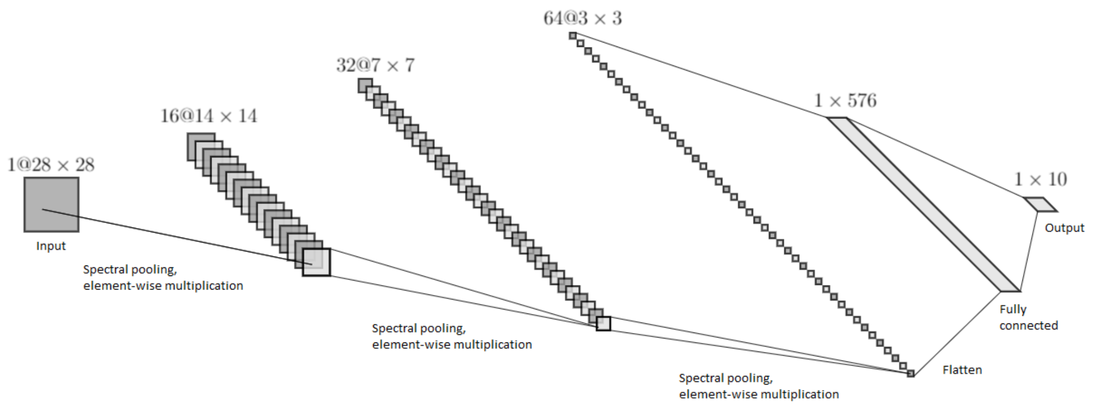


Figure 2.2: The schema of the proposed CNN architecture. The input is in the frequency domain and the spectral pooling can be done before the element-wise multiplication, and the nonlinear activation function can be applied after each multiplication.

## 2.4 Results and discussions

I started from a simple convolutional neural network and my goal was to implement all operations in the frequency domain after the initial Fourier transform and replace each element of the network with a suitable spectral solution, then 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 (Fig. 2.2) (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 2.1.

### 2.4.1 One-dimensional datasets

In this case, in the frequency domain and in the time domain, my network contained 3 convolutional layers and one FC (fully connected) layer. I performed one-dimensional convolution in the time domain with $8 \times 1$ kernels.

A one-dimensional dataset selected for detailed investigation was the Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set Version 2.1 (HADB, [77]). This consists of a smartphone's accelerometer and gyroscope signals during twelve different activities (such as standing, walking, walking downstairs and upstairs, laying, etc.) of 30 subjects. The training set contains more than 7,700 samples, while the test set contains 3,100 samples. The accuracy results for the spatial and spectral domain trainings are shown in Fig. 2.3.

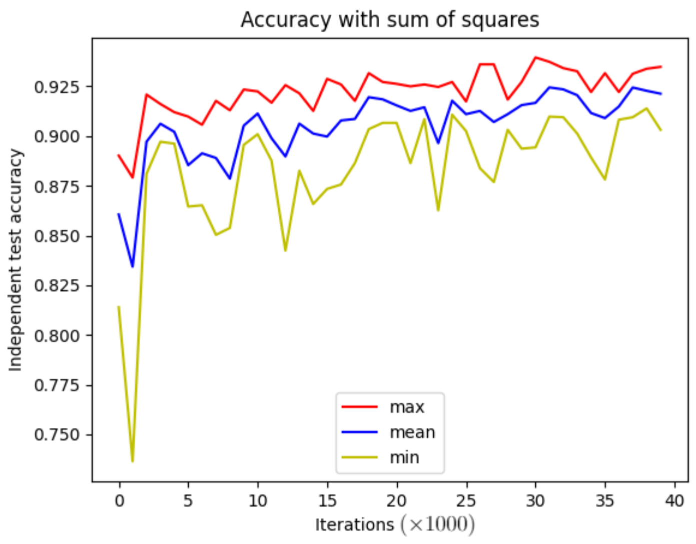


Figure 2.3: 5 training results of HADB classification in frequency domain with the proposed non-linear activation function, with square sum. The red color means the maximum accuracy, the yellow color is the minimum accuracy and the blue line shows the mean accuracy.

Another one-dimensional dataset was the Ozone Level Detection Data Set ([78]), I used the one hour peak set from that. The samples contain wind speed values at various time and temperature values measured at different times as well. These samples can be categorized into two classes, the first one is the normal day and the second one is the ozone day. The dataset has 2536 instances and I selected the last 500 as an independent test set. The results of this dataset are presented in Table 2.1.

### 2.4.2 Two-dimensional datasets

For two-dimensional datasets, in the frequency domain and in the time domain as well, I used a network with 3 convolutional layers and one FC (fully connected) layer. I performed two-dimensional convolution in the time domain with $3 \times 3$ kernels.

I used the well-known MNIST dataset, which is a database of handwritten digits, and it has a training set of 60,000 examples and a test set of 10,000 examples. The size of the images is 28x28. [79]

Another two-dimensional dataset was the Fashion-MNIST, which is an MNIST-like fashion product database with 10 classes and it consists of 28x28 sized greyscale images, where the number of elements of the training set is 60,000 and the test set has 10,000 examples. [80]

The Fig. 2.4 and Fig. 2.5 show the accuracy results of the independent test sets in the case of these datasets.

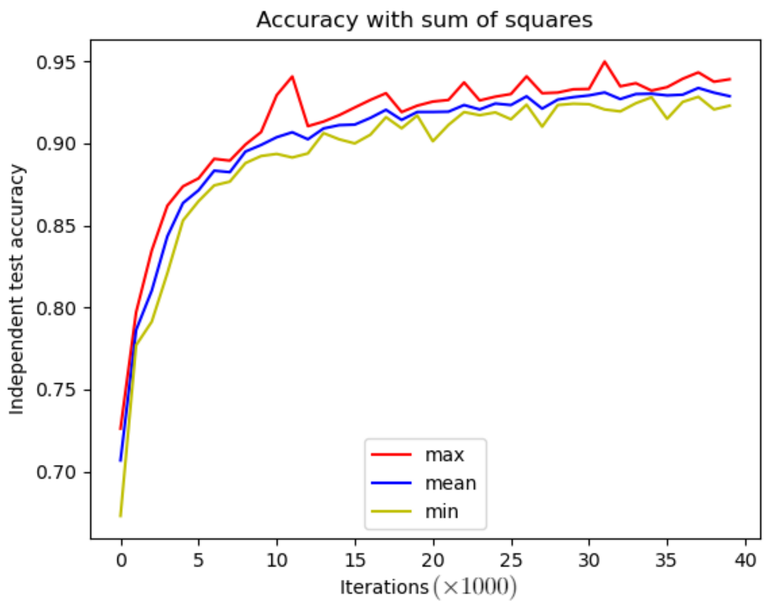


Figure 2.4: 5 training results of MNIST classification in frequency domain with the proposed non-linear activation function, with square sum instead of inverse FFT. The red color means the maximum accuracy, the blue line shows the mean accuracy and the yellow color is the minimum accuracy.

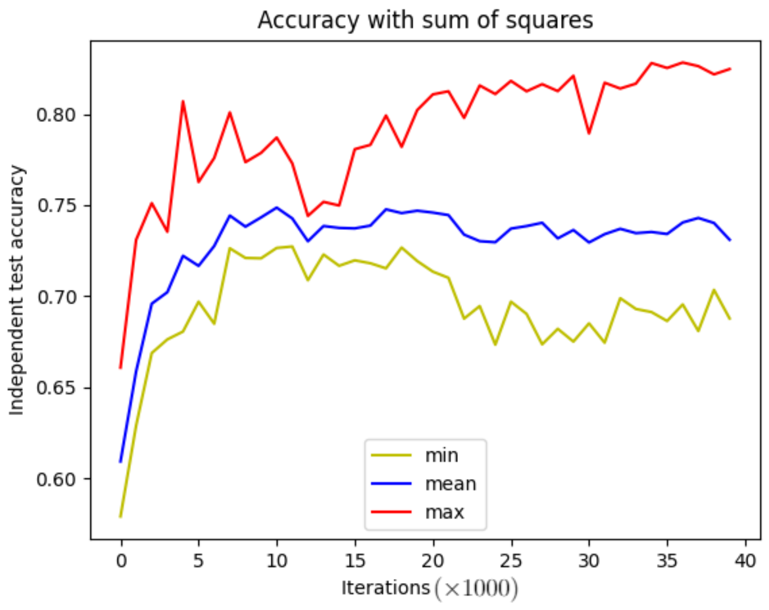


Figure 2.5: 5 training results of Fashion-MNIST classification in frequency domain with the proposed non-linear activation function, with square sum. The red color means the maximum accuracy, the yellow color is the minimum accuracy and the blue line shows the mean accuracy.

Table 2.1: 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 us.

| | 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%** |

Table 2.2: The table contains the number of multiplication in case of frequency-based implementation and in case of the time domain implementation. For example a typical input 224x224x3x3 number of multiplications is: 12544 in Fourier domain and 451584 in the time domain.

| | sum of squares | time domain |
|---|---|---|
| size of input | $(N/2) \times (M/2)$ | $N \times M$ |
| size of kernel | $(N/2) \times (M/2)$ | $H \times W$ |
| number of multiplications | $(N/2) \times (M/2)$ | $N \times M \times H \times W$ |

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 2.1) 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 (Table 2.2), 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.

### 2.4.3 Dependence on hyperparameters

On one hand my method is theoretical and it is easy to see that an end-to-end training in the Fourier domain can decrease the number of operations applied. These decrease is general and constantly present, since it comes from the re-formulation of the convolutions operation and the substitution of the inverse

Fourier transform. Because of this it can be applied and is advantageous for arbitrary network architectures and training processes. On the other hand the other important property, the accuracy of the investigated neural networks can not be determined theoretically and is typically measured empirically on commonly investigated datasets.

I have measured the performance of six different neural networks variants trained both in the time and Fourier domains and compared their performances. The results can be seen in Table 2.3. As it can be seen in the results the overall performance of a network depends on the selected hyperparameters, but the networks trained in the Fourier domain has always performed similarly as they time domain counterparts. The drop of the mean accuracy was 2% in average over the six investigated hyperparameter sets.

Table 2.3: The table contains the results of different hyperparameters both in the time and Fourier domains (without inverse FFT) on the HADB dataset. L denotes the number of channels in the consecutive layers from the first to last layer, opt denotes the optimizer which could be either Adam or Stochastic Gradient Descent (SGD).

| | Fourier domain (sum of squares) | | time domain | |
|---|---|---|---|---|
| hyperparameters | mean | max | mean | max |
| L: 16, 32; opt: Adam | **92.06%** | **95.02%** | 93.56% | 96.63% |
| L: 16 and 32; opt: SGD | 90.28% | 93.12% | 91.42% | 94.5% |
| L: 16, 32 and 64; opt: Adam | 91.93% | 94.99% | **97.17%** | 98.75% |
| L: 16, 32 and 64; opt: SGD | 91.4% | 94.26% | 89.3% | 93.21% |
| L: 32, 32 and 64; opt: Adam | 91.77% | 94.9% | 97.12% | **98.85%** |
| L: 32, 32 and 64; opt: SGD | 89.61% | 93.06% | 89.3% | 93.57% |

## 2.5 Discussion

As it can be seen from the results presented in the previous chapter my approach can provide an efficient implementation for convolutional neural networks with a minor drop in accuracy.

According to my knowledge this is the first approach where the whole training process is implemented in the Fourier domain. Other methods used certain operations or selected layers which were implemented in the Fourier domain but they all contained an inverse Fourier transform which requires a fairly high number of operations, typically comparable with the number of operations in a layer. Our results also demonstrate that convolutions can be more efficiently used in Fourier domain, which is a well known fact in the community. I substitute the traditional maximum pooling with spectral pooling which utilizes a different tpye of dimension reduction which suits ideally for the Fourier domain. The novelty of my method is threefold: 1, I have demonstrated that instead of complicated polynomial approximation of the standard time domain ReLU one can use a similar approach in the frequency domain and just cut off certain frequencies in the network. 2, I applied the training of the convolutional kernels directly in the frequency domain. Which means that the kernels are not initialized and later transformed to the Fourier domain for computation, but in my approach directly the Fourier version of the kernels are initialized. 3, I substitute the final inverse Fourier transformation of all previously published methods in the literature with a simple magnitude calculation which can reduce the number of operations from $nlog(n)$ to $n$, where $n$ is the number of neurons in the logit layer.

I have demonstrated that my method is general. I have demonstrated its validity on four different datasets and I have investigated six different network architectures. The results were consistent in all cases. The accuracy of the end-to-end Fourier domain networks dropped slightly, typically with 4%, but one can save three quarter of the multiplications compared to the traditional time series implementations of the networks.

This mean that this approach might not be a viable optimization strategy in applications where accuracy is utmost important, for example in medical image processing or navigation with self-driving cars, but in case of various other methods where power consumption is more critical like recommendation systems or photo enhancement or classification in personal photo libraries, my method could provide a viable and significant decrease in computation for possible applications of edge computing.

## 2.6 Conclusion

In [7], a convolutional neural network in the frequency domain was presented 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 was compared with similar network implementation containing inverse Fourier transformation. The proposed framework could achieve similar or better accuracy without the computational cost of inverse Fourier transformation. For instance, in the case of MNIST, which is a commonly used and often cited dataset, the maximum accuracy of 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 (sum of squares) dropped just approximately 4%.

# Chapter 3

# Convolutional Neural Network with Wave-based Convolver

## 3.1 Introduction

Artificial neural networks (ANNs) have become a staple of machine learning and they are more and more frequently applied in embedded systems as well. To enable their application among other things in mobile devices the decrease of computational need and their power consumption has to be significantly reduced. This is crucial because mobile devices have significantly limited computational resources and battery life, and are often used for tasks that require real-time processing. Thus the goal is to create ANNs that are lightweight and efficient, making them well suited for deployment in mobile devices and other resource-constrained environments.

One possible way of enabling low-energy computation can be the employment of special-purpose hardware accelerators, which use the physics of waves to naturally compute convolution integrals. Such devices are known in the literature as surface acoustic waves (SAWs) where a SAW-based convolver uses the interference of two counter-propagating waves to perform a convolution by the physics of the system( [81]). Another implementation of the device uses spin waves [82] - which are, in fact, very amenable to low-power, on-chip implementation for such hardware accelerator [83].

However in a real physical system (such as a wave-based convolver) dissipation is unavoidable and waves (especially spin waves) will decay over distance. This will influence the performance and the usefulness of the convolver. The purpose of the paper is to evaluate the performance of the convolver in the presence of such decay.

In [8], I demonstrate and investigate a special kernel convolution, which can be the cornerstone of a wave-based convolver device, that can yield a fast and energy-efficient building block of a convolutional neural network without a significant decrease in test accuracy. With my simulations I also identify the most important factors such as the attenuation (decay) and its affect on classification accuracy on commonly investigated datasets. These results can help in the construction of a device by identifying constraints regarding attenuation with which the device can function with high accuracy.

Nowadays, Convolutional Neural Networks (CNNs) have become a crucial tool for solving various artificial intelligence problems, and have been proven to deliver state-of-the-art results across a wide range of applications. In particular, CNNs have shown great success in image processing, video analysis, and natural language processing tasks. These tasks typically involve large amounts of high-dimensional data, such as images and videos, or complex relationships between words and sentences in text data. CNNs have been successful in these tasks because of their ability to automatically learn hierarchical representations of the data, and to identify and use the most discriminative features for a given task. As a result, they have become the go-to solution for many researchers and practitioners in these fields, and continue to be an active area of research and development.

One area where CNNs are used very successfully is the development of self-driving cars, for example, the traffic signs recognition and identification [84, 85, 86], navigation [87] and 3D object recognition [88], agrarian object identification [89] or in the medical field - like ECG signal classification and prediction [90], diabetic retinopathy recognition [91], thyroid nodule diagnosis [92], lung pattern classification for interstitial lung diseases [93] - in which their application has appeared even in mobile phones and embedded devices [94].

The main and most energy consuming operation of these architectures is convolution, so the optimal implementation of this operation is extremely important.

In the case of mobile and embedded vision applications, energy-saving implementation is also an important consideration. Lightweight deep neural networks (like MobileNets [58], Xnor-Nets [95] and spiking neural networks [96] ) can be used to achieve a reduction in energy consumption. Another possibility is to use a special device (such as FPGA[97] or ASIC devices [98]), which is able to perform the given operation extremely efficiently, thus the architecture will be faster and energy consumption will decrease.

However, in the case of low-power devices, especially emerging and non-Boolean devices, which exploit analogue and nonlinear device characteristics for computation will not be completely ideal, since the nonlinear dynamics of the device implementing the convolution may also affect the operation itself. But this nonlinear phenomenon is not necessarily a disatvantage, as the neural network requires some nonlinear operation (typically nonlinear activations such as the Rectified Linear Unit (ReLU) [99] or Scaled Exponential Linear Unit (SeLU) [100] functions provide these characteristics in the architecture) and Wang et al. [101] have shown that nonlinearity can be included in the convolution.

The authors introduced the applications of kernel convolution (kervolution), which was used to approximate complex behaviors of human perception systems. The kervolution generalizes convolution via kernel functions and the authors demonstrated that Kervolutional Neural Networks (KNNs) can achieve higher accuracy and faster convergence than the baseline convolutional neural networks. The authors' work represents an important contribution to the field of CNNs, and the use of KNNs in real-world applications holds significant promise. [101]

Neural network models containing kervolution can be effectively used, among others in case of anomaly detection, time series classification [102] and in authorship attribution [103]. Furthermore kervolution can be combined with left and right projection layers, thanks to which this model (ProKNN [104]) can be even more effective in certain situations.

In spread-spectrum communications, the real-time surface acoustic wave (SAW) convolver devices have been known since long time. These convolvers were also applied in programmable matched filtering to improve the signal-to-noise ratio, which was one of the first application of surface acoustic wave devices and it is an important potential in many cases. [81]

For example, radar systems have been widely used this process, since it enables the range of the system to be enlarged, for a given peak power limitation. [105]

Similar to SAW devices, it is conceivable to implement convolution in which it is performed using spin-wave magnetic devices, which may allow for much lower energy consumption and computation at higher frequencies[106].

Spin-wave computing uses magnetic excitations for computations. The spin-wave majority gates are one of the most prominent device concepts in this field. Linear passive logic gates, which are based on spin-wave interference, are a technology that takes the most advantage of the wave computing paradigm and therefore hold the highest promise for future ultralow-power electronics. [83]

The spin-wave circuits can be embedded also in CMOS (complementary metal–oxide–semiconductor) circuits, and these complete functional hybrid systems may outperform conventional CMOS circuits, since amongst other things they promise ultralow-power operation. Nowadays the challenges of these spin-wave circuit systems are low-power signal restoration and efficient spin-wave transducers. [83]

Furthermore, several methods have been proposed and studied for the development of spin-wave multiplexers and demultiplexers to greatly increase the data transmission capacity and efficiency of spin-wave systems. [83]

Therefore, based on the factors described above, I introduced a kervolutional neural network, where the kervolution was implemented by surface acoustic waves and the nonlinearities of the kervolutions based on the characteristic functions of magnetic devices.

## 3.2 Methods

In this section, I propose 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 my 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.

### 3.2.1 One-dimensional 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.

### 3.2.2 Convolution

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:

$$f \star g = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{3.1}$$

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:

$$[f * g](t) = \int_{0}^{t} f(\tau)g(t-\tau)d\tau \tag{3.2}$$

This operation can be implemented by real-time SAW convolvers, such as three-port elastic SAW convolver (3.1) under nonlinear operation [81].

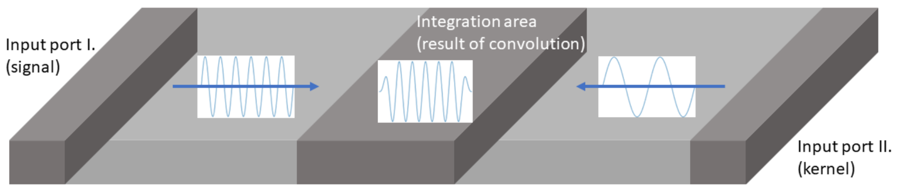


Figure 3.1: This figure depicts a primitive three-port elastic SAW convolver. Two signals travel in opposite directions form the two inputs ports of the device (at the left and right edges of the device) and the convolved version of the two signals can be extracted at the integration area (in the middle 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:

$$s(t,z) = S(t - z/\nu)e^{j(\omega_0 t - \beta z)} \tag{3.3}$$

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:

$$r(t,z) = R(t + z/\nu)e^{j(\omega_0 t - \beta z)} \tag{3.4}$$

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:

$$C(t) = P \int_{-\frac{L}{2}}^{+\frac{L}{2}} S(t - z/\nu) R(t + z/\nu) dz e^{j2\omega_0 t} \tag{3.5}$$

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:

$$C(t) = Mve^{j2\omega_0 t} \int_{-\infty}^{+\infty} S(\tau) R(2t - \tau) d\tau \tag{3.6}$$

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 $\omega_0$ is the angular frequency of the signal. [81]

The equations (3.1) and (3.6) 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. [81]

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 [83].

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 3.2. 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.

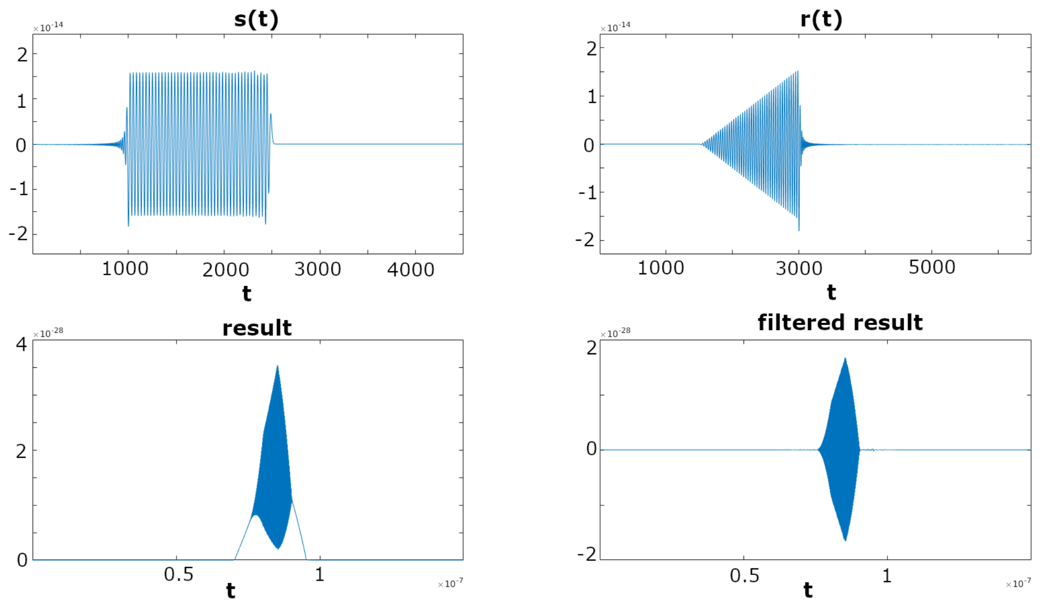


Figure 3.2: In the first row, $s(t)$ is the square signal, and next to this function is the triangular signal $r(t)$ (inverted in time), these are travelling opposite each other. The first plot in the second row is the raw result, which is read from the collision of the above signals. The last plot is the frequency filtered result, which has doubled frequency compared to the original signals.

#### 3.2.2.1 SAW based kervolution

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

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

$$g_i(x) =< \phi_i(x_i), \phi_i(w) > \tag{3.7}$$

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:

$$\phi_i(x) = e^{\frac{-i}{a}} x_i \tag{3.8}$$

where $i$ is the discrete time, $a$ is the attenuation parameter. Fig. 3.3 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.)

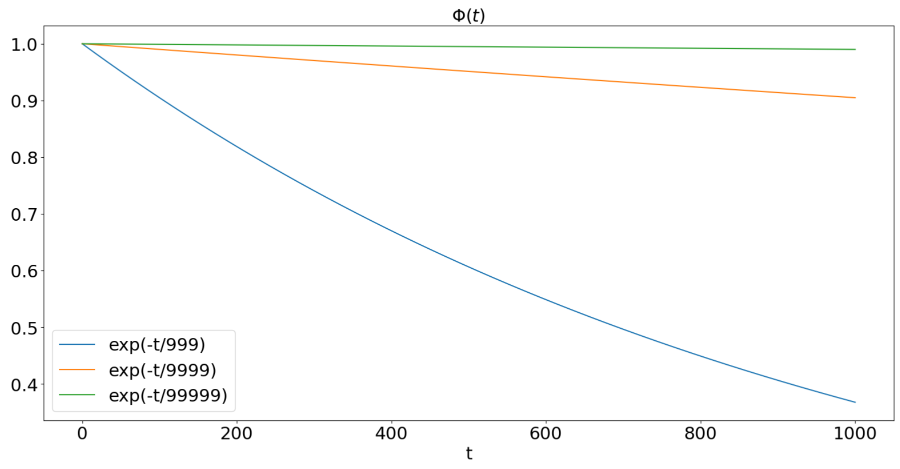


Figure 3.3: One of the parameters of my simulation is attenuation. The propagating waves were attenuated exponentially with the function $e^{\frac{-i}{a}}$, where $i$ is the time and $a$ is the attenuation parameter. Here I plot the function with 999, 9999 and 99999 attenuation parameters. Since the dependence of attenuation is exponential on this parameter it can heavily affect the accuracy of a neural network.

## 3.3 Results

I started from a simple convolutional neural network and my 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. 3.5) 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 $1 \times 9$ 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. 3.4. 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 3.1.

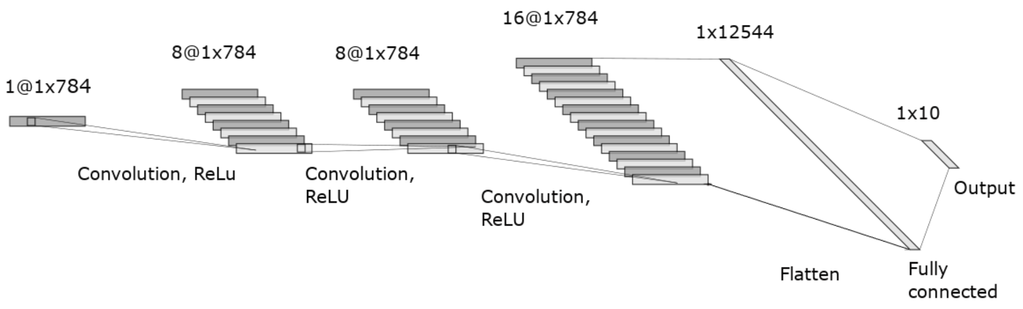


Figure 3.4: The architecture of my reference neural network model. My network contain three convolutional layers with ReLus followed by a fully connected layer. This simple four layered architecture is capable of solving simple classification tasks.

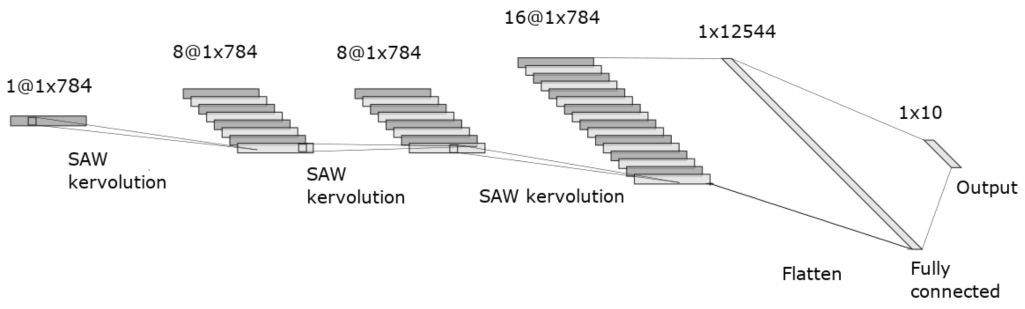


Figure 3.5: The architecture of my neural network model with SAW kervolution. The convolutions and ReLUs are substituted with kervolutions in this variants. Please note that the number of layers, channels and parameters are the same in both network variants.

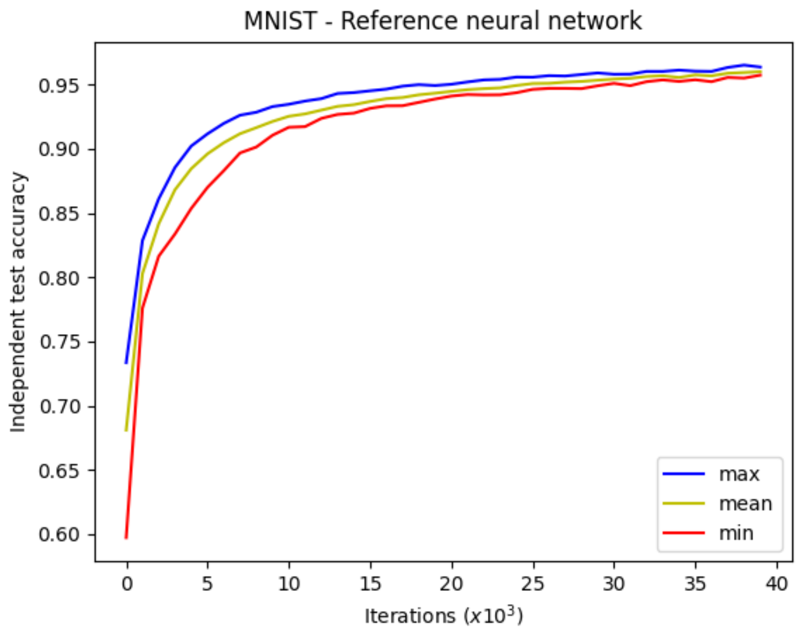
As a more complex two-dimensional case study I have investigated the well-known MNIST dataset as which contains handwritten digits, and it has a training set of 60,000 examples and a test set of 10,000 examples. The size of the images is 28x28 pixels. Another two-dimensional dataset was the Fashion-MNIST, which is an MNIST-like fashion product database with 10 classes and it consists of 28x28 sized greyscale images, where the number of elements of the training set is 60,000 and the test set has 10,000 examples. The evolution of the classification accuracies on the test set of the MNIST dataset with a traditional convolutional network and a kervolutional network implemented by an SAW convolver can be seen in Fig. 3.6 and the confusion matrices of the trained architectures can be found in Fig 3.9. The same results for Fashion-MNIST can be observed in Fig 3.7 and 3.10 accordingly.

I examined one-dimensional datasets as well. One of those is Ozone Level Detection Data Set ([78]), I used the one-hour peak set from that. The samples contain wind speed values at various moments and temperature values measured at different times as well. These samples can be categorized into two classes, the first one is the normal day and the second one is the ozone day class. The dataset has 2536 instances and I selected the last 500 as an independent test set. The classification accuracy results of this dataset can be found in Table 3.1 along with other accuracy results on the previously mentioned datasets.

As it can be seen from the results in this table 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. This demonstrates that an energy efficient SAW convolver could provide a viable implementation in certain problems where this 6% accuracy drop is acceptable.

Another examined one-dimensional database is the Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set Version 2.1 (HADB, [77]). This consists of a smartphone's accelerometer and gyroscope signals during twelve different activities (such as standing, walking, walking downstairs and upstairs, laying, etc.) of 30 subjects. The training set contains more than 7,700 samples, while the test set contains roughly 3,100 samples. The test accuracies on

this dataset during training are depicted in Fig. 3.8.



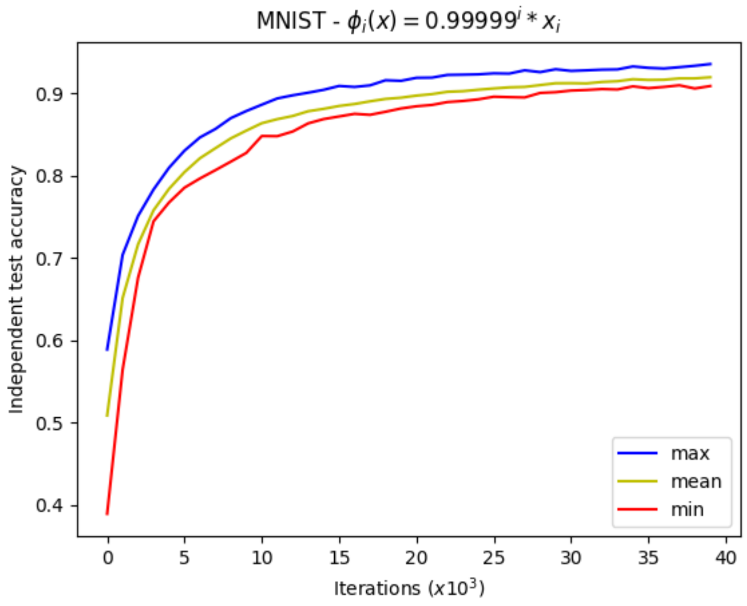
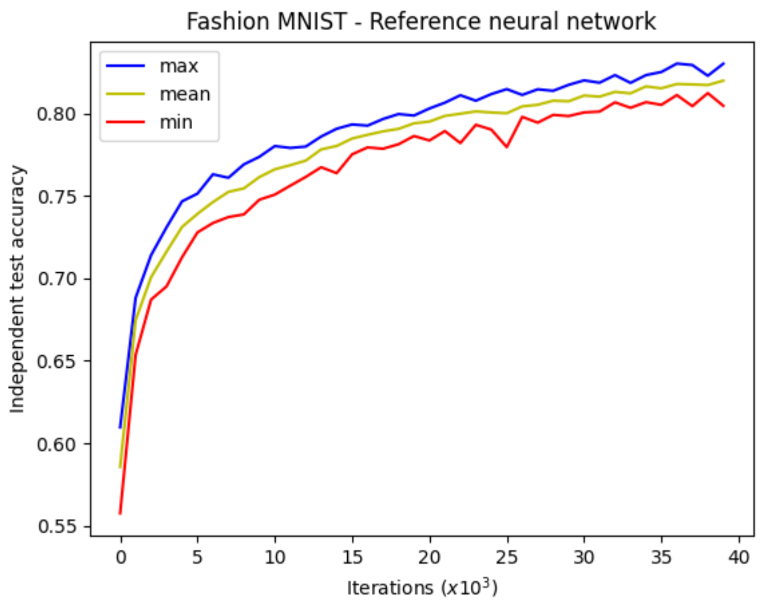
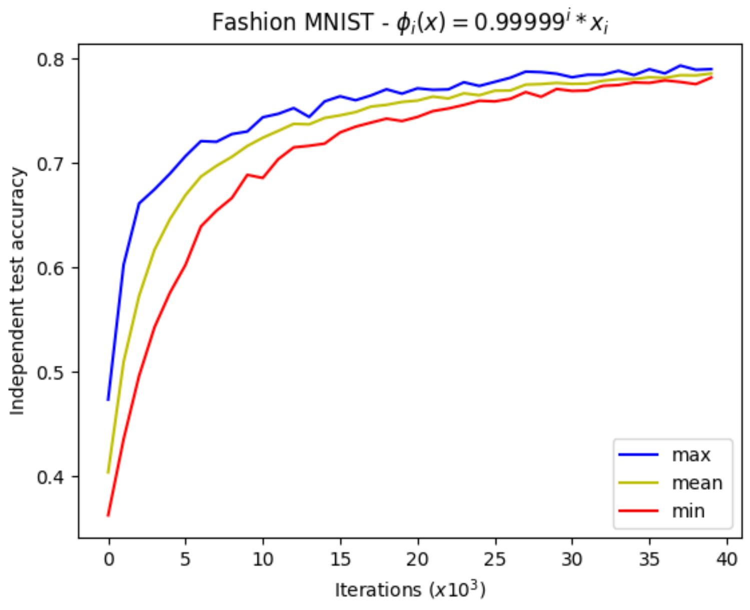


Figure 3.6: This figure plots the mean, minimal and maximal classification accuracies during training averaged from five independent training on the MNIST dataset. The top image (a) depicts the classification accuracies with the reference convolutional neural network (baseline), meanwhile the lower image (b) plots the same results using an SAW convolution based kervolutional neural network with an attenuation parameter of 99999. The blue color means the maximum accuracy, the yellow color is the mean accuracy and the red line shows the minimum accuracy. As it can be seen there are no significant differences between the two results, except for the 6% drop in performance

.



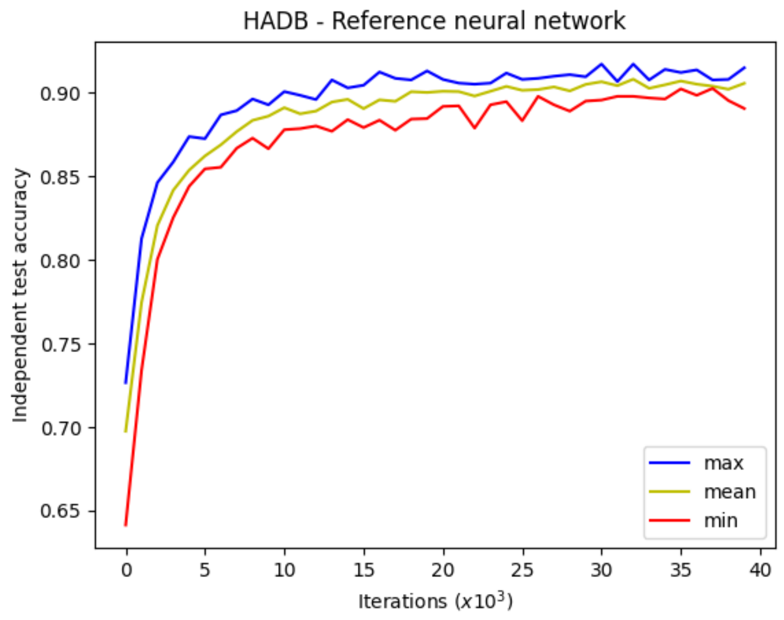
(a)



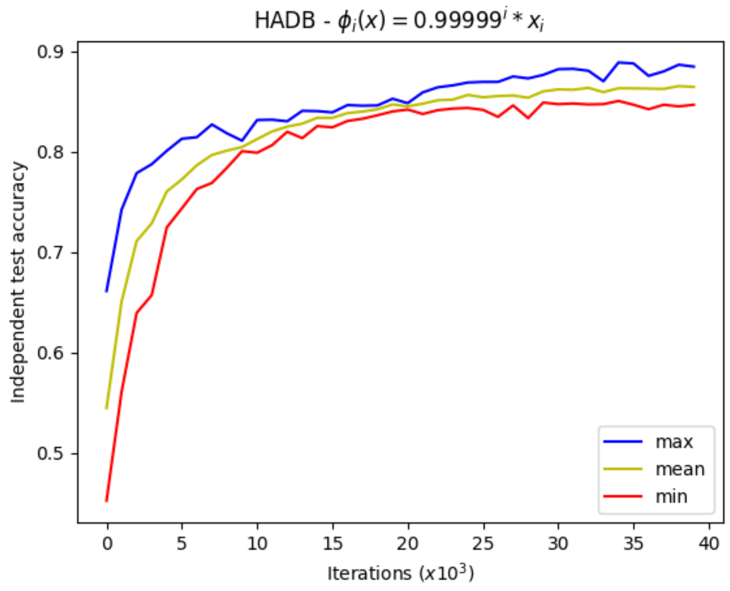
(b)

Figure 3.7: This figure plots the mean, minimal and maximal classification accuracies during training averaged from five independent training on the Fashion-MNIST dataset. The top image (a) depicts the classification accuracies with the reference convolutional neural network (baseline), meanwhile the lower image (b) plots the same results using an SAW convolution based kervolutional neural network with an attenuation parameter of 99999. The blue color means the maximum accuracy, the yellow color is the mean accuracy and the red line shows the minimum accuracy. As it can be seen there are no significant differences between the two results, except for the 5% drop in performance

.



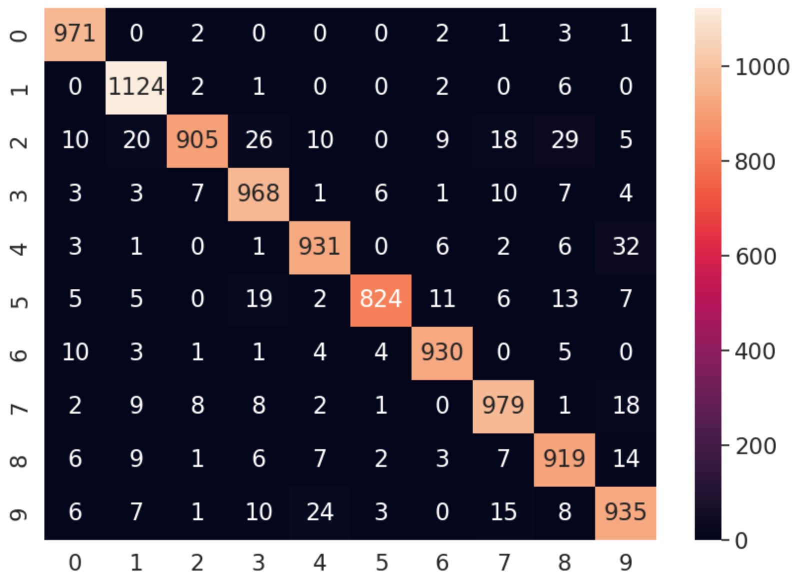
(a)



(b)

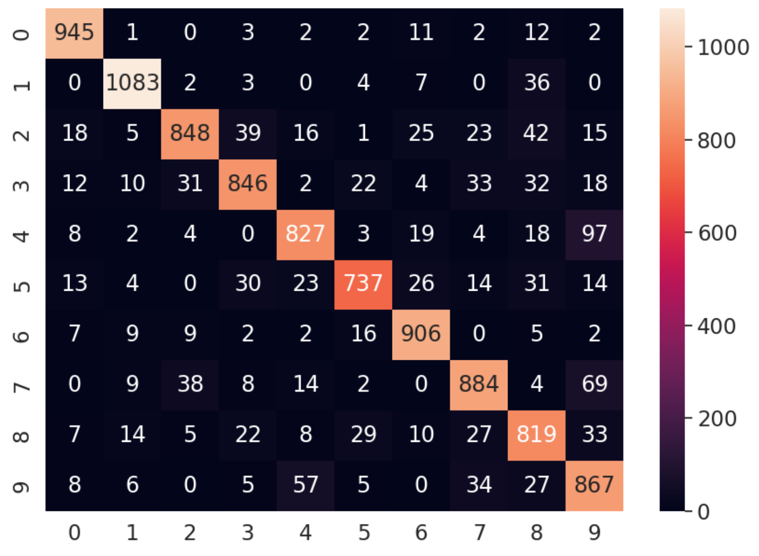
Figure 3.8: This figure plots the mean, minimal and maximal classification accuracies during training averaged from five independent training on the HADB dataset. The top image (a) depicts the classification accuracies with the reference convolutional neural network (baseline), meanwhile the lower image (b) plots the same results using an SAW convolution based kervolutional neural network with an attenuation parameter of 99999. The blue color means the maximum accuracy, the yellow color is the mean accuracy and the red line shows the minimum accuracy. As it can be seen there are no significant differences between the two results, except for the 6% drop in performance

.



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 971 | 0 | 2 | 0 | 0 | 0 | 2 | 1 | 3 | 1 |
| 1 | 0 | 1124 | 2 | 1 | 0 | 0 | 2 | 0 | 6 | 0 |
| 2 | 10 | 20 | 905 | 26 | 10 | 0 | 9 | 18 | 29 | 5 |
| 3 | 3 | 3 | 7 | 968 | 1 | 6 | 1 | 10 | 7 | 4 |
| 4 | 3 | 1 | 0 | 1 | 931 | 0 | 6 | 2 | 6 | 32 |
| 5 | 5 | 5 | 0 | 19 | 2 | 824 | 11 | 6 | 13 | 7 |
| 6 | 10 | 3 | 1 | 1 | 4 | 4 | 930 | 0 | 5 | 0 |
| 7 | 2 | 9 | 8 | 8 | 2 | 1 | 0 | 979 | 1 | 18 |
| 8 | 6 | 9 | 1 | 6 | 7 | 2 | 3 | 7 | 919 | 14 |
| 9 | 6 | 7 | 1 | 10 | 24 | 3 | 0 | 15 | 8 | 935 |

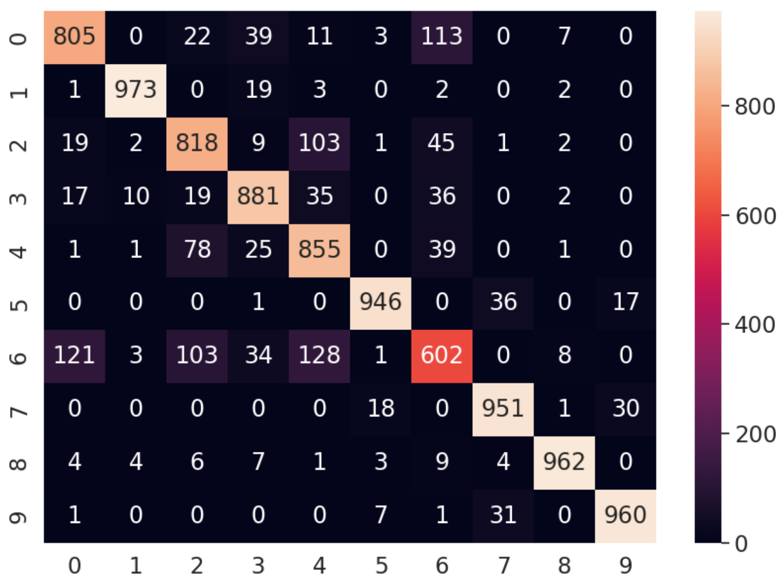
(a)



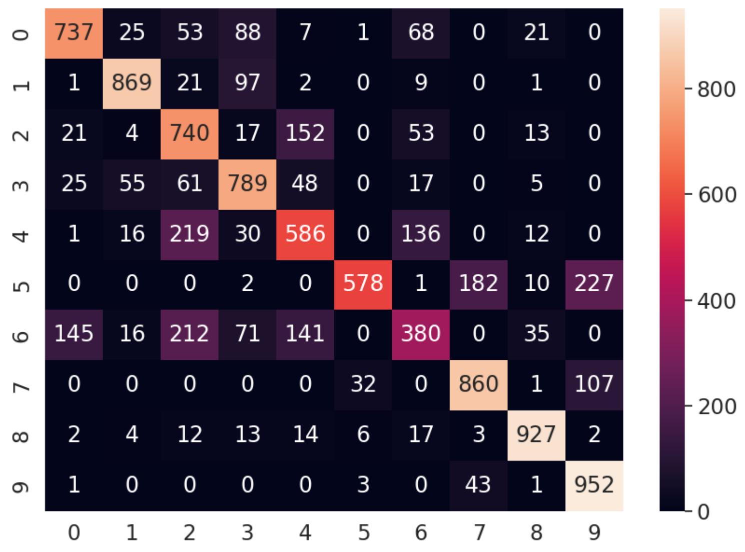
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 945 | 1 | 0 | 3 | 2 | 2 | 11 | 2 | 12 | 2 |
| 1 | 0 | 1083 | 2 | 3 | 0 | 4 | 7 | 0 | 36 | 0 |
| 2 | 18 | 5 | 848 | 39 | 16 | 1 | 25 | 23 | 42 | 15 |
| 3 | 12 | 10 | 31 | 846 | 2 | 22 | 4 | 33 | 32 | 18 |
| 4 | 8 | 2 | 4 | 0 | 827 | 3 | 19 | 4 | 18 | 97 |
| 5 | 13 | 4 | 0 | 30 | 23 | 737 | 26 | 14 | 31 | 14 |
| 6 | 7 | 9 | 9 | 2 | 2 | 16 | 906 | 0 | 5 | 2 |
| 7 | 0 | 9 | 38 | 8 | 14 | 2 | 0 | 884 | 4 | 69 |
| 8 | 7 | 14 | 5 | 22 | 8 | 29 | 10 | 27 | 819 | 33 |
| 9 | 8 | 6 | 0 | 5 | 57 | 5 | 0 | 34 | 27 | 867 |

(b)

Figure 3.9: The confusion matrix of a trained architecture on the test set of the MNIST dataset, in case of the reference CNN model (a) and in case of my kervolutional neural network model (b). As one can see the two confusion matrices are qualitatively similar and although there are more misclassification in the case of the SAW convolver (which is natural since it has a slightly lower accuracy), the distribution of the errors is similar.

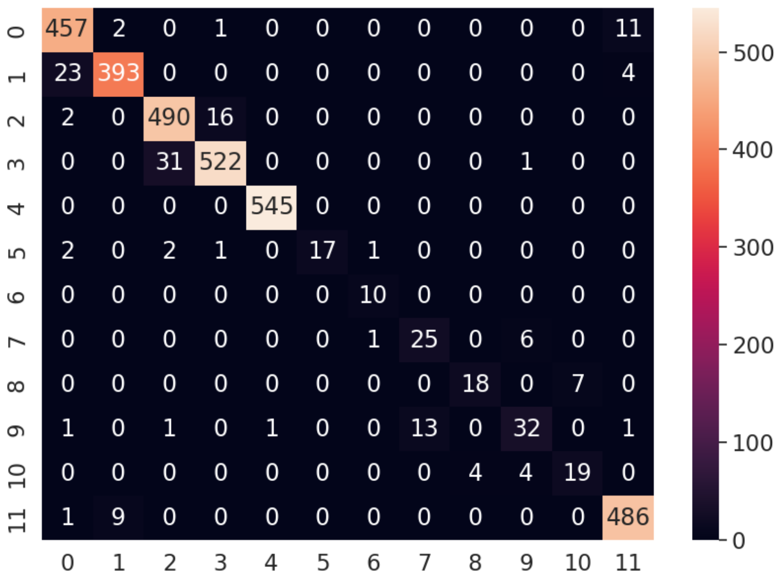


(a)



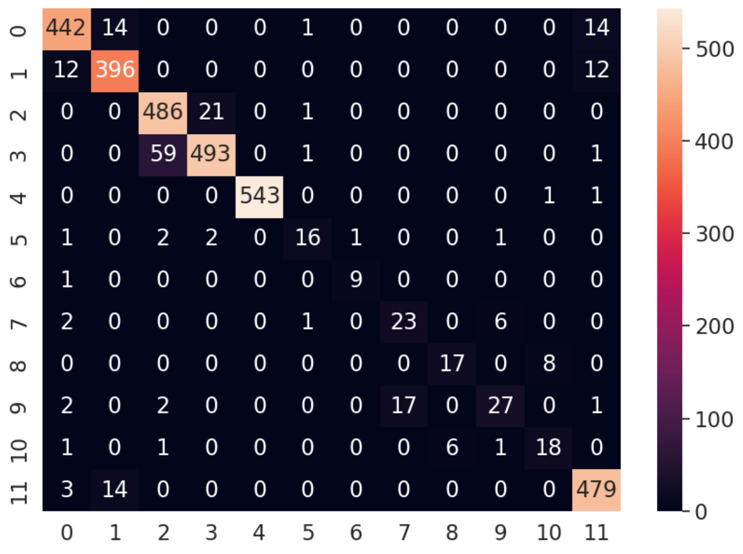
(b)

Figure 3.10: The confusion matrix of a trained architecture on the test set of the Fashion-MNIST dataset, in case of the reference CNN model (a) and in case of my kervolutional neural network model (b). As one can see the two confusion matrices are qualitatively similar and although there are more misclassification in the case of the SAW convolver (which is natural since it has a slightly lower accuracy), the distribution of the errors is similar. (The number labels mean: 0: T-shirt, 1: Trouser, 2: Pullover, 3: Dress, 4: Coat, 5: Sandal, 6: Shirt, 7: Sneaker, 8: Bag, 9: Ankle boot.)



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 457 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 |
| 1 | 23 | 393 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 2 | 2 | 0 | 490 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 31 | 522 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 545 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 2 | 0 | 2 | 1 | 0 | 17 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 25 | 0 | 6 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18 | 0 | 7 | 0 |
| 9 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 13 | 0 | 32 | 0 | 1 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 19 | 0 |
| 11 | 1 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 486 |

(a)



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 442 | 14 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 14 |
| 1 | 12 | 396 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| 2 | 0 | 0 | 486 | 21 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 59 | 493 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 543 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 5 | 1 | 0 | 2 | 2 | 0 | 16 | 1 | 0 | 0 | 1 | 0 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 |
| 7 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 23 | 0 | 6 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 | 0 | 8 | 0 |
| 9 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 17 | 0 | 27 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 6 | 1 | 18 | 0 |
| 11 | 3 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 479 |

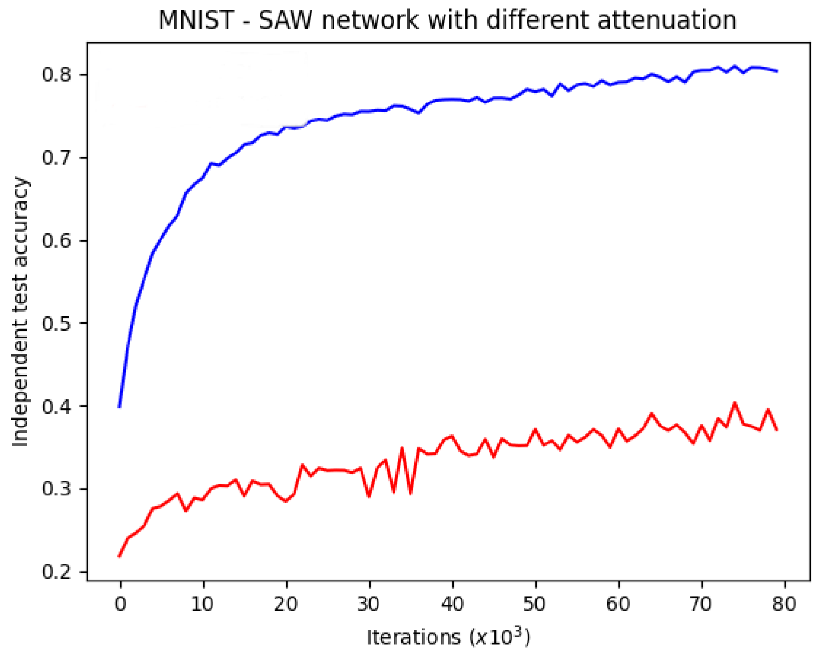
(b)

Figure 3.11: The confusion matrix of a trained architecture on the test set of the HADB dataset, in case of the reference CNN model (a) and in case of my kervolutional neural network model (b). As one can see the two confusion matrices are qualitatively similar and although there are more misclassification in the case of the SAW convolver (which is natural since it has a slightly lower accuracy), the distribution of the errors is similar. (The number labels mean: 0: walking, 1: walking upstairs, 2: walking downstairs, 3: sitting, 4: standing, 5: laying, 6: stand to sit, 7: sit to stand, 8: sit to lie, 9: lie to sit, 10: stand to lie, 11: lie to stand.)

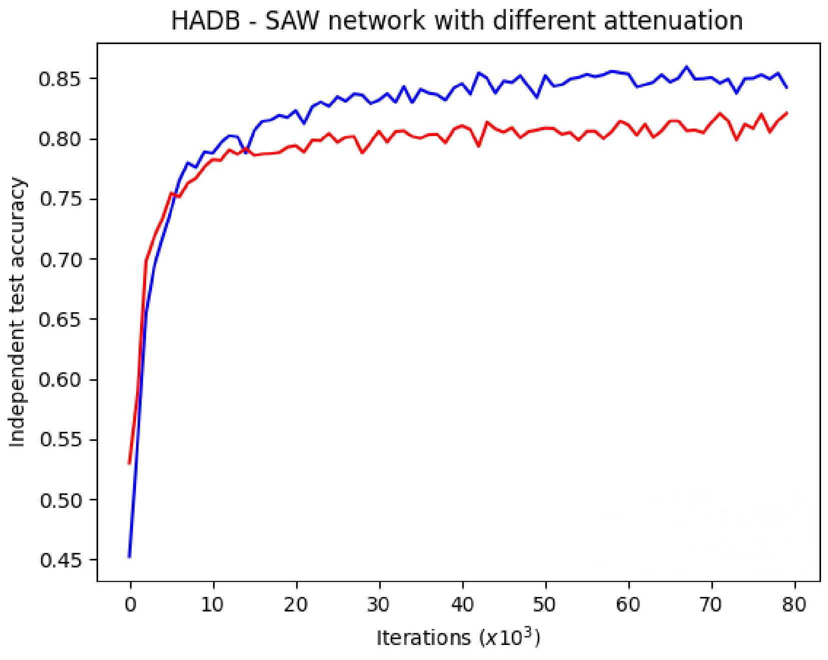
| | Reference network | | network with SAW kervolution | |
|---|---|---|---|---|
| Dataset | mean | max | mean | max |
| MNIST | 92.61% | **96.52%** | 86.51% | 93.58% |
| Fashion-MNIST | 77.84% | **83.01%** | 72.87% | 79.32% |
| HADB | 88.43% | **91.71%** | 82.11% | 88.89% |
| OZONE | 99.15% | 99.2%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

Table 3.1: 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. The test accuracies for both datasets can be seen in Fig. 3.12. As these plots demonstrate if the attenuation parameter ($a$) is larger than 9999 the network reaches similar accuracy as reported in Table 3.1 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.



(a)



(b)

Figure 3.12: The training results of MNIST (a) and HADB (b) classification with different $a$ (attenuation) parameters, in case of blue curves the $a$ is 9999 and red curves show the result of 999 parameter.

## 3.4 Conclusion

In [8], 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 implementation containing normal convolution. The network accuracy was decreased with an average of 5% in case of kervolutions, but these operations can enable low-energy implementation on embedded devices. The proposed framework could achieve similar or a little worse accuracy, but it has the potential to be implemented with a much faster and more energy-efficient device. Our results have also revealed some of the required properties of future magnetic devices. To ensure high accuracy the attenuation parameter cannot be lower than $e^{\frac{-i}{999}}$.

In [8], I used a very simple convolutional neural network architecture to examine the capability of my kernel convolution. Thus in the future, I want to investigate more complex and sophisticated network architecture that can be applied to a wide range of real-world problems in various fields. Also, a more accurate understanding of the physical parameters and the practical implementation of the physical device are also important parts of my future plans. To achieve these objectives, I plan to conduct further research and experimentation in these areas.

# Chapter 4

# Machine learning for the design of k-space magnetization dynamics

In contemporary computational research, nonlinear magnetization dynamics have emerged as a promising avenue for developing unconventional computing devices. Significantly, magnetic devices are positioning themselves as prominent candidates in the area of non-Boolean and neuromorphic computing architectures.

However, a persistent challenge in this domain is the crafting of magnonic devices that not only align with theoretical principles but also deliver tangible computational utility.

While simulating magnetization dynamics may appear straightforward, designing these computing instruments demands the resolution of inverse problems. Specifically, there's a need to delineate optimal nanomagnet geometries and associated external field excitations that effectively address distinct computational challenges.

In this discussion, I introduce computational methodologies optimized for developing neuromorphic computing devices. These devices leverage the potent, non-linear excitations of a Yttrium Iron Garnet (YIG) disc. When analyzed, these excitations manifest as oscillatory modes, derived from micromagnetic simulations via Dynamic Mode Decomposition (DMD).

While DMD has gained traction in modeling complex nonlinear processes, its application to magnetization dynamics remains relatively novel. My exploration into this synergy reveals that nanomagnet behaviors can be accurately represented using approximately $n = 10$ distinct modes. As we escalate in amplitude, interactions among DMD modes become pronounced, requiring external fields to modulate these interactions adeptly.

To shape these crucial programming waveforms, one can employ advanced machine learning techniques or delve into the intricacies of the temporal variations of DMD amplitudes.

In this chapter, I present a strategy for designing outer space waveforms aimed at increasing computational efficiency.

## 4.1 The k-space computing

In artificial neural networks the neurons receive multiple inputs, compute a weighted superposition of them, and once a nonlinear operation is applied to this superposition, the neural output is passed to next layers. This methodology is intuitive when the system is designed as an electrical circuit. In cases where neuromorphic computing is achieved through a dynamic physical mechanism, like a magnonic setup, then the computational framework is different from the standard neuron-based concept.

I used a method that links neural operations to the oscillatory modes of a nanomagnet. Data processing is governed by the time dynamics of the modal amplitudes. Within this apparatus, two concurrent signals are introduced to the magnet: an evolving input signal and a similarly time-variant programming signal. The computational result is denoted by the amplitude of the normal mode. Instead of establishing a network in real space, calculations are conducted in the domain of wavevector-space, known as k-space. Although this diverges significantly from conventional neural networks, the neural network in k-space can be trained via gradient-based techniques, similar to those used in conventional neural network architectures.

### 4.1.1 Training in k-space

In training a dynamic system, the forward path involves solving the ordinary differential equations (ODEs) that describe the system. This is done using a time-marching procedure, such as the Runge-Kutta method, with all intermediate results being saved. The backward differentiation step involves a gradient-based optimization that adjusts the physical parameters in a direction that minimizes the loss function.

Magnetization dynamics can be described using a normal mode representation. In this approach, rather than using the discretized $M$ distribution, the time evolution of the $a_i(t)$ mode amplitudes is determined. This method is especially relevant when the magnetization dynamics can be represented as a superposition of a limited number of eigenmodes. Importantly, the number of significant normal modes corresponds to the system's degrees of freedom. [107]

Normal modes can be identified using frequency-domain micromagnetic methods. Once these calculations and evaluations of mode couplings are conducted for a specific geometry (in my simulation, the magnet was a rectangular-shaped plate, which can also be seen in the Fig. 4.1), the normal mode description still applies to arbitrary small-amplitude RF field excitations. [107]

## 4.2 Simulation

During my simulation, I based my model on a magnetic architecture that has two line source inputs. One of these inputs processes the signal to be analyzed, while the other introduces a programming signal. This programming signal is configured (with the aid of a machine learning process) to ensure that the architecture provides the correct output for various signals. The system's output is a specific magnetic value measured at a particular time and averaged over a certain area of the magnet. A simplified figure (Fig. 4.1) depicts the architecture.

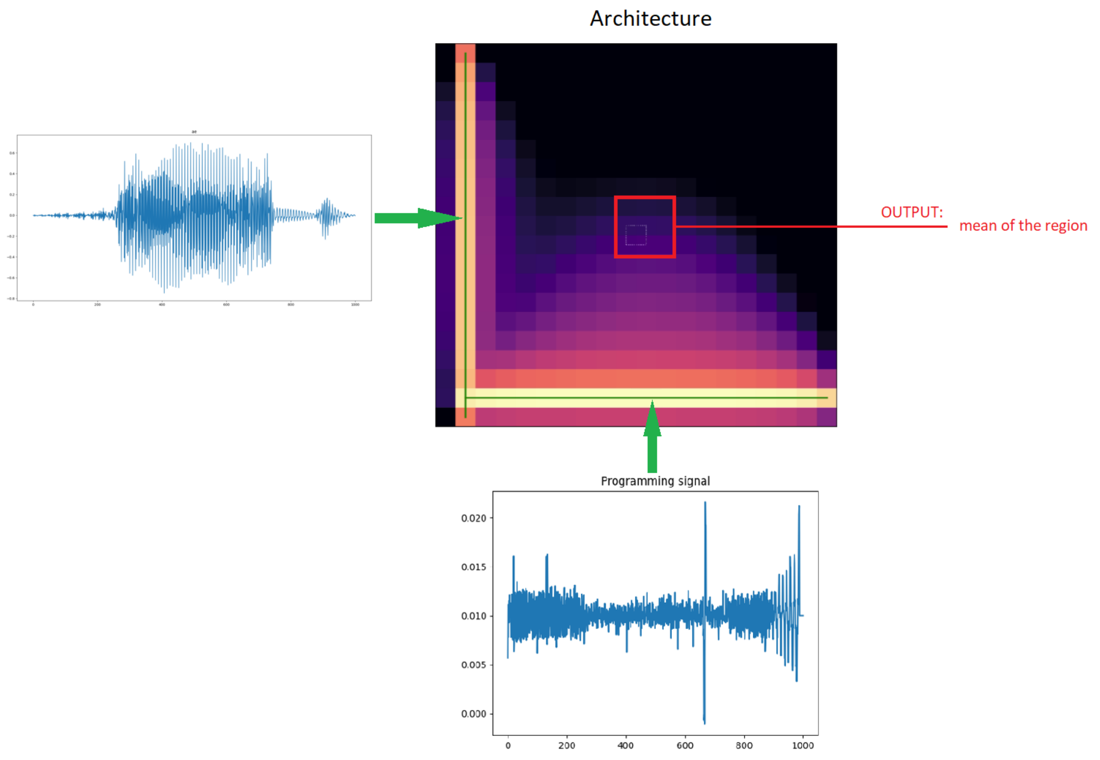


Figure 4.1: The architecture of my simulation. My architecture has two line source inputs, one of these inputs processes the signal to be analyzed, while the other introduces a programming signal. The system's output is a specific magnetic value measured at a particular time and averaged over a certain area of the magnet.

During my simulation (which was implemented in Python and I used Pytorch for the machine learning process), the main physical parameters I used are listed in the Table 4.1. Such parameters include the discretization parameters (dx, dy, dz); the number of cells in the x-y direction (nx, ny); the saturation magnetization (Ms in A/m); the bias field ($B_0$ in Tesla); excitation field amplitude ($B_t$ in Tesla); timestep (dt - in seconds); source frequency ($f_1$ in Hz); and the number of timesteps for wave propagation.

Table 4.1: The table contains the parameters of my simulation.

| parameter | value |
|---|---|
| dx | 50e-9 (m) |
| dy | 50e-9 (m) |
| dz | 20e-9 (m) |
| nx | 30 |
| ny | 30 |
| Ms | 140e3 (A/m) |
| $B_0$ | 60e-2 (T) |
| $B_t$ | 5e-4 (T) |
| $dt$ | 3e-12 (s) |
| $f_1$ | 4e9 (Hz) |
| number of timesteps | 4e3 |

### 4.2.1 Dataset

In my experiment, the dataset on which I verified the architecture's learnability and the efficiency of the learning consisted of audio files. Specifically, these were vowels recorded in American English. For simplicity, given that I was only validating the concept of a basic experimental architecture, I considered just two vowels. This meant there were vowels belonging to two distinct classes that the system needed to distinguish between: 'ae' and 'ah'. I had a total of 24-24 audio files for each class, and I used their formants as inputs to the system. The formants utilized for each sample included steady state, 20%, 50%, and 80% value formants. More precisely, the input was linear combinations of sinusoidal signals for a given sample, and the frequencies of the sine waves corresponded to these formant values. An example of such an input signal can be seen in the Fig. 4.2.

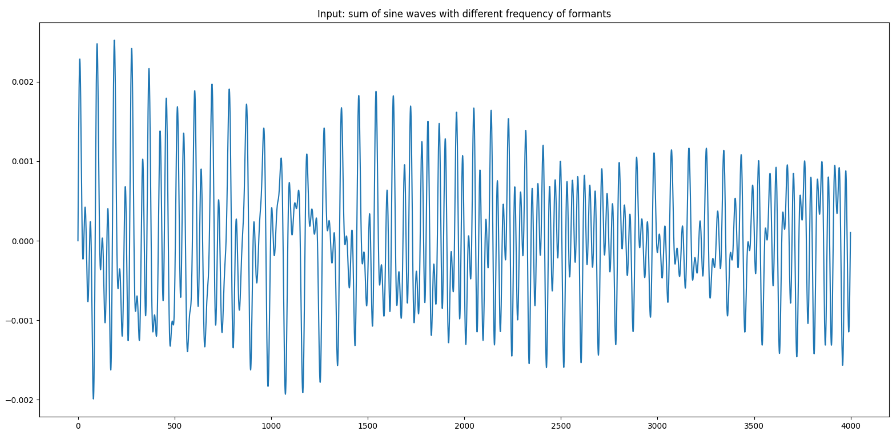


Figure 4.2: An example input of my simulation. I used linear combinations of sinusoidal signals, with the frequencies of the sine waves corresponding to the formant values from the 'ae' and 'ah' vowel audio files.

### 4.2.2 Training of the architecture

During the training of my model, I trained the programming signal input of the architecture. My goal was for the system's output (an averaged amplitude over a specified area, serving as a temporal signal at the output) to be different for the two classes. I determined the distinctiveness of the output signal by applying a Fourier transformation and then assessing which of two pre-selected and fixed frequency ranges had the larger integral of the FFT transformed signal. Thus, the classification was determined by which integral was larger in the frequency domain: either the one within the range indicated by the black vertical lines (seen in the Fig. 4.3) or the one in the range marked in magenta.

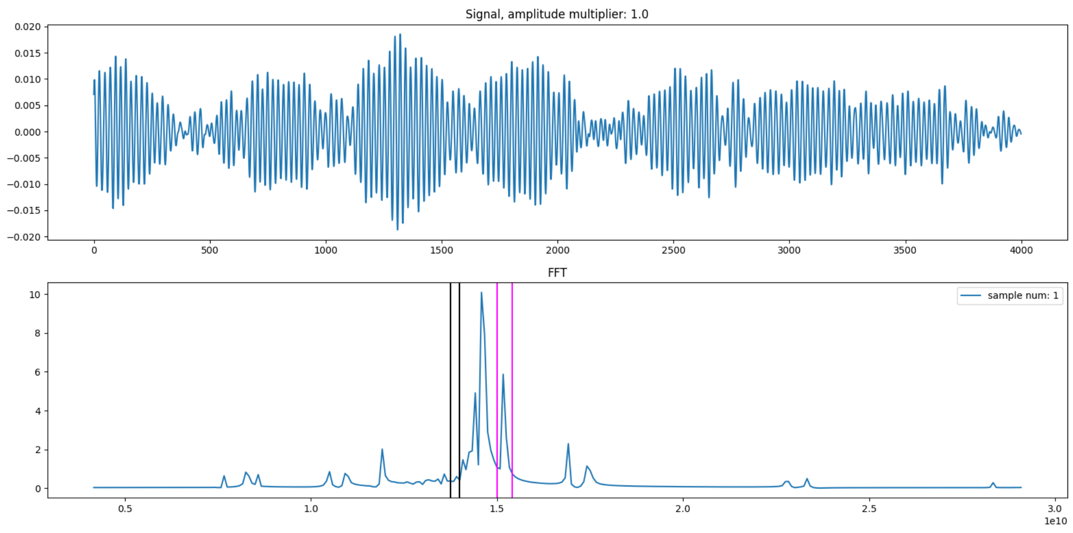


Figure 4.3: An example output of my simulation. The classification was determined by which integral was larger in the frequency domain: either the one within the range indicated by the black vertical lines or the one in the range marked in magenta.

## 4.3 Results

During the training, the learning parameter was the frequency of the programming sinusoidal signal. Its fine-tuning was carried out over a total of 10 epochs until the system was trained. Since the complete dataset consisted of 48 samples, I randomly selected 10 samples from it (5 samples per class), which represented the independent test set, while the remaining 38 samples comprised the training set on which the system learned. After 10 epochs, the trained model performed flawlessly on the independent set, correctly classifying all 10 samples into their respective groups. The Fig. 4.4 illustrates how the accuracy evolved during the training for the training set. The Table 4.2 contains the training information.

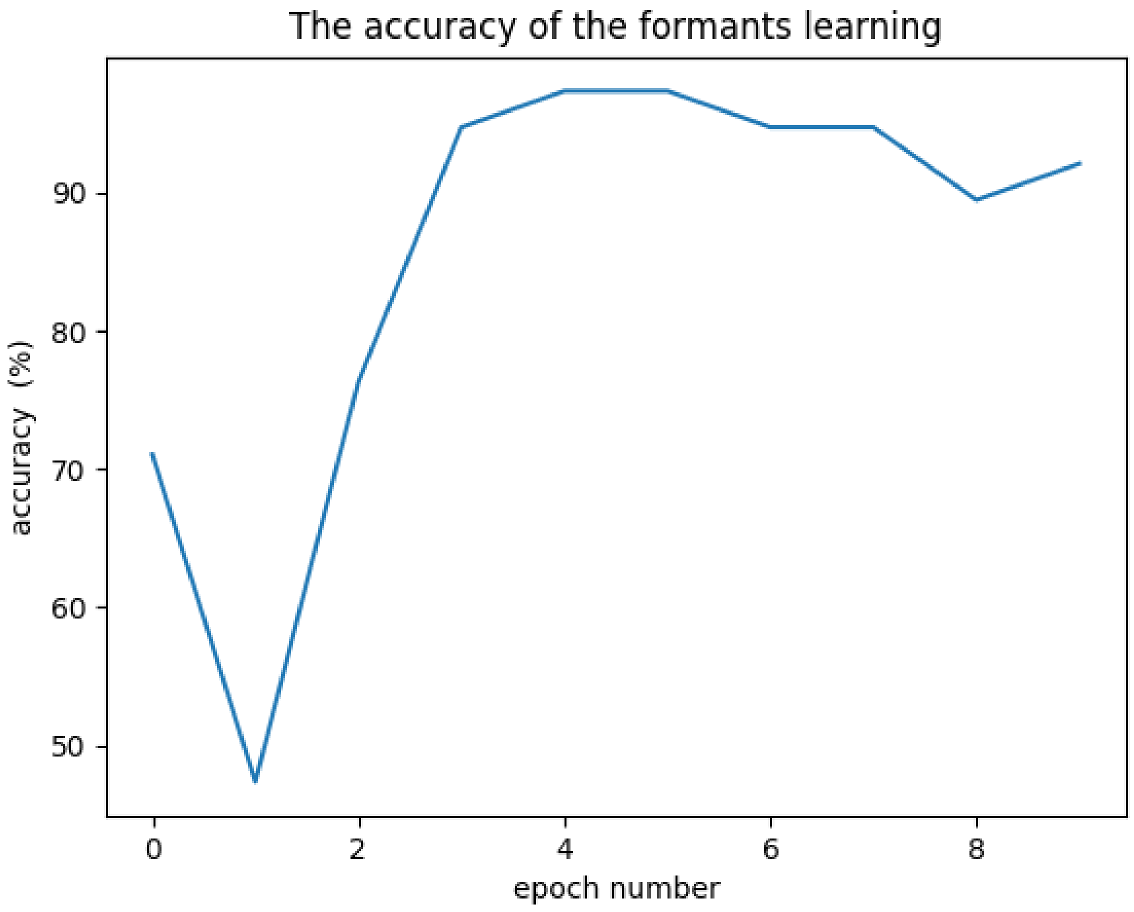


Figure 4.4: This plot illustrates how the accuracy evolved during the training for the training set.

Table 4.2: The table contains the parameters of simulation.

| parameter | value |
|---|---|
| training dataset | 38 samples |
| test dataset | 10 samples (m) |
| number of training epoch | 10 |
| test set accuracy | 100% |

## 4.4 Conclusion

In this chapter, utilizing the combination of machine learning and the temporal DMD amplitude changes, I developed a new method, during which I conducted a simulation as an experiment. By using audio files, specifically recognizing the difference between the vowels 'ae' and 'ah', I demonstrated through my architecture. The flawless classification after the short training phase showcased that the architecture could be effective for certain signal classification tasks.

I hope that the presented results not only supplement the current knowledge in the field but also prove valuable for future research in neuromorphic computing devices.

# Chapter 5

# Summary of the dissertation

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

## 5.1 Frequency Domain CNNs

The first approach focuses on implementing the entire training process in the frequency domain without using any inverse Fourier transformation. By introducing an alternative realization of spatial activation functions in the frequency domain, we can achieve 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 approach can be especially valuable for applications where energy efficiency and computational speed are critical.

## 5.2 Wave-Based Convolver CNNs

The second part of the dissertation introduces a special CNN architecture with a wave-based convolver inspired by physical devices like Surface Acoustic Wave (SAW) convolvers. The idea is to leverage the physical properties of such devices to perform convolution efficiently.

The implementation of the wave-based convolver includes exponential attenuation and saturation effects in the system, which are characteristics of the simulated physical device.

Instead of using classical nonlinear activation functions like ReLU or sigmoid, the system incorporates nonlinearity through the physical properties (characteristics) of the simulated device.

Experiments were conducted on various datasets, comparing the performance of the wave-based convolver CNNs with traditional CNNs. The results show that the wave-based convolver CNNs achieve similar accuracy to traditional CNNs, with a small drop in most cases. However, the wave-based convolver implementation can offer energy-efficient solutions for specific applications and it opens up possibilities for low-energy implementations.

## 5.3 Conclusion

Both approaches presented in this dissertation offer new ways to implement CNNs with potential benefits in terms of energy efficiency and computational speed.

Overall, these novel approaches contribute to the advancement of CNN architectures and pave the way for energy-efficient and faster implementations of deep learning models.

# Reference

[1] O. E. David, N. S. Netanyahu, and L. Wolf, “Deepchess: End-to-end deep neural network for automatic learning in chess,” in *International Conference on Artificial Neural Networks*, pp. 88–96, Springer, 2016. 1

[2] C. Clark and A. Storkey, “Training deep convolutional neural networks to play go,” in *International conference on machine learning*, pp. 1766–1774, PMLR, 2015. 1

[3] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, “A method to estimate the energy consumption of deep neural networks,” in *2017 51st asilomar conference on signals, systems, and computers*, pp. 1916–1920, IEEE, 2017. 1

[4] L. F. W. Anthony, B. Kanding, and R. Selvan, “Carbontracker: Tracking and predicting the carbon footprint of training deep learning models,” *arXiv preprint arXiv:2007.03051*, 2020. 1

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020. 1, 1.3.2.1

[6] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, “Carbon emissions and large neural network training,” *arXiv preprint arXiv:2104.10350*, 2021. 1

[7] A. Fülöp and A. Horváth, “End-to-end training of deep neural networks in the fourier domain,” *Mathematics*, vol. 10, no. 12, p. 2132, 2022. 1.1, 2.6

[8] A. Fülöp, G. Csaba, and A. Horváth, "A convolutional neural network with a wave-based convolver," *Electronics*, vol. 12, no. 5, p. 1126, 2023. 1.1, 3.1, 3.4

[9] H. Wang and B. Raj, "A survey: Time travel in deep learning space: An introduction to deep learning models and how deep learning models evolved from the initial ideas," *arXiv preprint arXiv:1510.04781*, 2015. 1.3.1

[10] H. Wang and B. Raj, "On the origin of deep learning," *arXiv preprint arXiv:1702.07800*, 2017. 1.3.1, 1.3.1

[11] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943. 1.3.1

[12] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958. 1.3.1

[13] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*, 1974. 1.3.1

[14] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982. 1.3.1

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 1.3.1

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018. 1.3.1

[17] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2022. 1.3.1

[18] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022. 1.3.1

[19] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, *et al.*, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, no. 9, p. 999, 2021. 1.3.1

[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006. 1.3.1

[21] C. Wehmeyer and F. Noé, "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics," *The Journal of chemical physics*, vol. 148, no. 24, 2018. 1.3.1

[22] Y. Kiarashinejad, S. Abdollahramezani, and A. Adibi, "Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures," *npj Computational Materials*, vol. 6, no. 1, p. 12, 2020. 1.3.1

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 1.3.1

[24] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2020. 1.3.1

[25] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008. 1.3.1

[26] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022. 1.3.1

[27] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012. 1.3.1

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. 1.3.1

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS'2014*, 2014. 1.3.1

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015. 1.3.1

[31] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in neural information processing systems*, vol. 30, 2017. 1.3.1

[32] P. Afshar, A. Mohammadi, and K. N. Plataniotis, "Brain tumor type classification via capsule networks," in *2018 25th IEEE international conference on image processing (ICIP)*, pp. 3129–3133, IEEE, 2018. 1.3.1

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. 1.3.1

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 1.3.1, 1.3.2.1

[35] A. Ettinger, "What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 34–48, 2020. 1.3.1

[36] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009. 1.3.1

[37] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2337–2348, 2022. 1.3.1

[38] M. A. Gordon, K. Duh, and N. Andrews, "Compressing bert: Studying the effects of weight pruning on transfer learning," *arXiv preprint arXiv:2002.08307*, 2020. 1.3.1

[39] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019. 1.3.2.1

[40] Y. Cui, W. Che, T. Liu, B. Qin, S. Wang, and G. Hu, "Revisiting pre-trained models for chinese natural language processing," *arXiv preprint arXiv:2004.13922*, 2020. 1.3.2.1

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015. 1.3.2.2

[42] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017. 1.3.2.2

[43] P. Lakhani and B. Sundaram, "Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks," *Radiology*, vol. 284, no. 2, pp. 574–582, 2017. 1.3.2.2

[44] J. Zhang, M. Liu, and D. Shen, "Detecting anatomical landmarks from limited medical imaging data using two-stage task-oriented deep neural networks," *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 4753–4764, 2017. 1.3.2.2

[45] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, pp. 3–11, Springer, 2018. 1.3.2.2

[46] T. Hirasawa, K. Aoyama, T. Tanimoto, S. Ishihara, S. Shichijo, T. Ozawa, T. Ohnishi, M. Fujishiro, K. Matsuo, J. Fujisaki, *et al.*, "Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images," *Gastric Cancer*, vol. 21, pp. 653–660, 2018. 1.3.2.2

[47] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016. 1.3.2.3

[48] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. 1.3.2.3

[49] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A general pipeline for 3d detection of vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3194–3200, IEEE, 2018. 1.3.2.3

[50] S. Chadwick, W. Maddern, and P. Newman, "Distant vehicle detection using radar and vision," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8311–8317, IEEE, 2019. 1.3.2.3

[51] L. Zhao, Y. Liu, A. Y. Al-Dubai, A. Y. Zomaya, G. Min, and A. Hawbani, "A novel generation-adversarial-network-based vehicle trajectory prediction method for intelligent vehicular networks," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 2066–2077, 2020. 1.3.2.3

[52] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017. 2.1

[53] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, “Cvt: Introducing convolutions to vision transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22–31, 2021. 2.1

[54] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, “Do vision transformers see like convolutional neural networks?,” *Advances in Neural Information Processing Systems*, vol. 34, 2021. 2.1

[55] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020. 2.1

[56] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, “Repvgg: Making vgg-style convnets great again,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13733–13742, 2021. 2.1

[57] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016. 2.1

[58] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017. 2.1, 3.1

[59] J. Yim, D. Joo, J. Bae, and J. Kim, “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–4141, 2017. 2.1

[60] A. Farhadi and M. Rastegari, “System and methods for efficiently implementing a convolutional neural network incorporating binarized filter and convolution operation for performing image classification,” Sept. 19 2019. US Patent App. 16/430,123. 2.1

[61] Y. Liu and M. K. Ng, “Deep neural network compression by tucker decomposition with nonlinear response,” *Knowledge-Based Systems*, p. 108171, 2022. 2.1

[62] Y.-D. Kim and S. Choi, “Nonnegative tucker decomposition,” in *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2007. 2.1

[63] S.-C. Hsia, S.-H. Wang, and C.-Y. Chang, “Convolution neural network with low operation flops and high accuracy for image recognition,” *Journal of Real-Time Image Processing*, vol. 18, no. 4, pp. 1309–1319, 2021. 2.1

[64] Y. Wang, M. Huang, K. Han, H. Chen, W. Zhang, C. Xu, and D. Tao, “Addernet and its minimalist hardware design for energy-efficient artificial intelligence,” *arXiv preprint arXiv:2101.10015*, 2021. 2.1

[65] M. Mathieu, M. Henaff, and Y. LeCun, “Fast training of convolutional networks through ffts,” *arXiv preprint arXiv:1312.5851*, 2013. 2.1, 2.2

[66] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, “Fast convolutional nets with fbfft: A gpu performance evaluation,” *arXiv preprint arXiv:1412.7580*, 2014. 2.1, 2.2

[67] J. Lin, L. Ma, and Y. Yao, “A fourier domain training framework for convolutional neural networks based on the fourier domain pyramid pooling method and fourier domain exponential linear unit,” *IEEE Access*, vol. 7, pp. 116612–116631, 2019. 2.1, 2.2, 2.3.2.2

[68] S. O. Ayat, M. Khalil-Hani, A. A.-H. Ab Rahman, and H. Abdellatef, “Spectral-based convolutional neural network without multiple spatial-frequency domain switchings,” *Neurocomputing*, vol. 364, pp. 152–167, 2019. 2.1, 2.2, 2.3.2.2, 2.3.2.2

[69] T. Watanabe and D. F. Wolf, "Image classification in frequency domain with 2srelu: a second harmonics superposition activation function," *arXiv preprint arXiv:2006.10853*, 2020. 2.1, 2.2

[70] A. S. Kushchenko, N. E. Ternovoy, M. G. Popov, and A. N. Yakunin, "Implementation of convolution function through fast fourier transform in convolution neural networks computation," in *2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pp. 368–370, IEEE, 2022. 2.1

[71] Y. Xu and H. Nakayama, "Dct-based fast spectral convolution for deep convolutional neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2021. 2.1

[72] B. Sun, W. Xu, and Q. Yang, "Analog circuit fault diagnosis based on fft-cnn-lstm," in *2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, pp. 305–310, IEEE, 2021. 2.1

[73] T. Highlander and A. Rodriguez, "Very efficient training of convolutional neural networks using fast fourier transform and overlap-and-add," *arXiv preprint arXiv:1601.06815*, 2016. 2.2

[74] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in neural information processing systems*, pp. 2449–2457, 2015. 2.2, 2.3.2.3

[75] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "Fcnn: Fourier convolutional neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 786–798, Springer, 2017. 2.2

[76] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *arXiv preprint arXiv:1412.6830*, 2014. 2.3.2.2

[77] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016. 2.4.1, 3.3

[78] D. Dua and C. Graff, "UCI machine learning repository," 2017. 2.4.1, 3.3

[79] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010. 2.4.2

[80] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. 2.4.2

[81] C. Campbell, *Surface acoustic wave devices and their signal processing applications*. Elsevier, 2012. 3.1, 3.2.2, 3.2.2

[82] V. I. Vasyuchka, G. A. Melkov, V. A. Moiseienko, A. V. Prokopenko, and A. N. Slavin, "Correlation receiver of below-noise pulsed signals based on parametric interactions of spin waves in magnetic films," *Journal of magnetism and magnetic materials 321, no. 20 : 3498-3501*, vol. 2, 2009. 3.1

[83] F. V. VMahmoud Abdulqader, Florin Ciubotaru, "Introduction to spin wave computing.," *Journal of Applied Physics 128, no. 16*, vol. 2, 2020. 3.1, 3.2.2

[84] W. Farag, "Recognition of traffic signs by convolutional neural nets for self-driving vehicles," *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 22, no. 3, pp. 205–214, 2018. 3.1

[85] W. Li, D. Li, and S. Zeng, "Traffic sign recognition with a small convolutional neural network," in *IOP Conference Series: Materials Science and Engineering*, vol. 688, p. 044034, IOP Publishing, 2019. 3.1

[86] H. H. Aghdam, E. J. Heravi, and D. Puig, "A practical and highly optimized convolutional neural network for classifying traffic signs in real-time," *International Journal of Computer Vision*, vol. 122, no. 2, pp. 246–269, 2017. 3.1

[87] T.-D. Do, M.-T. Duong, Q.-V. Dang, and M.-H. Le, "Real-time self-driving car navigation using deep neural network," in *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, pp. 7–12, IEEE, 2018. 3.1

[88] R. D. Singh, A. Mittal, and R. K. Bhatia, "3d convolutional neural network for object recognition: a review," *Multimedia Tools and Applications*, vol. 78, no. 12, pp. 15951–15995, 2019. 3.1

[89] Ł. Chechliński, B. Siemiątkowska, and M. Majewski, "A system for weeds and crops identification—reaching over 10 fps on raspberry pi with the usage of mobilenets, densenet and custom modifications," *Sensors*, vol. 19, no. 17, p. 3787, 2019. 3.1

[90] W. Caesarendra, T. A. Hishamuddin, D. T. C. Lai, A. Husaini, L. Nurhasanah, A. Glowacz, and G. A. F. Alfarisy, "An embedded system using convolutional neural network model for online and real-time ecg signal classification and prediction," *Diagnostics*, vol. 12, no. 4, p. 795, 2022. 3.1

[91] H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding, and Y. Zheng, "Convolutional neural networks for diabetic retinopathy," *Procedia computer science*, vol. 90, pp. 200–205, 2016. 3.1

[92] J. Ma, F. Wu, J. Zhu, D. Xu, and D. Kong, "A pre-trained convolutional neural network based method for thyroid nodule diagnosis," *Ultrasonics*, vol. 73, pp. 221–230, 2017. 3.1

[93] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, "Lung pattern classification for interstitial lung diseases using a deep convolutional neural network," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1207–1216, 2016. 3.1

[94] H. Li, H. Wang, L. Liu, and M. Gruteser, "Automatic unusual driving event identification for dependable self-driving," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pp. 15–27, 2018. 3.1

[95] A. Bulat and G. Tzimiropoulos, “Xnor-net++: Improved binary neural networks,” *arXiv preprint arXiv:1909.13863*, 2019. 3.1

[96] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, “Spiking neural networks and their applications: A review,” *Brain Sciences*, vol. 12, no. 7, p. 863, 2022. 3.1

[97] R. Zhao, H.-C. Ng, W. Luk, and X. Niu, “Towards efficient convolutional neural network for domain-specific applications on fpga,” in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 147–1477, IEEE, 2018. 3.1

[98] A. Boutros, S. Yazdanshenas, and V. Betz, “You cannot improve what you do not measure: Fpga vs. asic efficiency gaps for convolutional neural network inference,” *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018. 3.1

[99] S. Khalife and A. Basu, “Neural networks with linear threshold activations: structure and algorithms,” in *International Conference on Integer Programming and Combinatorial Optimization*, pp. 347–360, Springer, 2022. 3.1

[100] I. Jahan, M. F. Ahmed, M. O. Ali, and Y. M. Jang, “Self-gated rectified linear unit for performance improvement of deep neural networks,” *ICT Express*, 2022. 3.1

[101] C. Wang, J. Yang, L. Xie, and J. Yuan, “Kervolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 31–40, 2019. 3.1

[102] O. Ammann, G. Michau, and O. Fink, “Anomaly detection and classification in time series with kervolutional neural networks,” *arXiv preprint arXiv:2005.07078*, 2020. 3.1

[103] C. Suman, A. Raj, S. Saha, and P. Bhattacharyya, “Authorship attribution of microtext using capsule networks,” *IEEE Transactions on Computational Social Systems*, 2021. 3.1

[104] M. Mulimani, R. Nandi, and S. G. Koolagudi, “Acoustic scene classification using projection kervolutional neural network,” *Multimedia Tools and Applications*, pp. 1–11, 2022. 3.1

[105] D. Morgan, *Surface acoustic wave filters: With applications to electronic communications and signal processing.* Academic Press, 2010. 3.1

[106] R. Sasaki, Y. Nii, and Y. Onose, “Magnetization control by angular momentum transfer from surface acoustic wave to ferromagnetic spin moments,” *Nature communications*, vol. 12, no. 1, pp. 1–7, 2021. 3.1

[107] G. Csaba, Á. Papp, H. András, J.-V. Kim, M. Massouras, A. Anane, M. d’Aquino, S. Perna, and C. Serpico, “Design of k-space magnon dynamics by machine learning,” in *2023 IEEE International Magnetic Conference-Short Papers (INTERMAG Short Papers)*, pp. 1–2, IEEE, 2023. 4.1.1