Univerza v Ljubljani
Fakulteta za matematiko in fiziko

Complex-valued convolutional neural networks

Seminar I - 1. letnik, II. stopnja

Author
Matic Debeljak
Supervisor
doc. dr. Simon Čopar

Ljubljana, 2023

Abstract

This seminar explores the Complex-Valued Convolutional Neural Networks (CvCNNs) and their application in various domains. We begin by reviewing the foundational concepts of Artificial Neural Networks (ANNs) and their successes in classification tasks. We continue with Convolutional Neural Networks (CNNs) and their ability to capture spatial patterns in images. Finally, we will introduce CvCNNs as an extension to handle complex-valued data. We will show their potential in the field of magnetic resonance imaging reconstruction.

Contents

1 Introduction

Computers have traditionally been used to solve problems that are difficult for human beings, but can be easily described by explicit rules or formulas. However, they have faced challenges when it comes to tasks that require intuition or pattern recognition, such as image or speech recognition.

In the 1940s, inspired by the structure and function of the human brain, artificial neural networks (ANNs) were built, then known as "cybernetics" [1]. The initial models allowed for training of a single neuron. However, it wasn't until the 1980s that scientists achieved a significant advancement with a "connectionist" [1] approach using back-propagation, which allowed for the training of neural networks with hidden layers.

Despite these breakthroughs, the limited computational power available at the time made training large networks slow and computationally expensive, leading many researchers to question their usefulness. Additionally, neural networks in the 1980s were typically trained on small and simplistic datasets due to a lack of data, which made it difficult to evaluate their performance on more complex tasks.

Another breakthrough began in 2006 with faster computers and bigger datasets. These developments have enabled significant progress in the field of neural networks and have opened up new possibilities for solving previously to demanding problems.

However, traditional neural networks are limited by their inability to handle complex data structures such as graphs or sequences, and by their dependence on real-valued inputs and outputs. Complex valued neural networks address some of these limitations by using complex numbers.

In this seminar, we will delve into the fundamentals of neural networks and then focus on the modifications and benefits introduced by complex-valued neural networks. By incorporating complex-valued representations, complex-valued convolutional neural networks (CvNNs)

offer a promising approach to tackle more challenging tasks and improve performance in various domains.

2 Neural networks

An artificial neural network consists of an input layer, hidden layers, and an output layer. Each neuron receives input from other neurons or external data sources, processes that information using an activation function (f) and generates an output signal that is passed to the next layer of neurons or the output layer. This process is referred to as feedforward, which describes the flow of information.

$$\mathbf{a}^{(l)} = f\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{1}$$

where $\mathbf{a}$ represents the vector of activations, $\mathbf{W}$ denotes the weight matrix, $\mathbf{b}$ represents the biases vector and l represents the layer number.

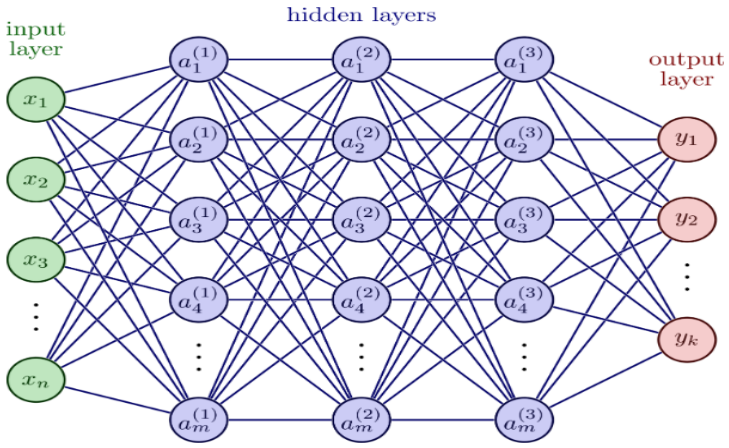

Figure 1: Architecture of neural network with 3 hidden layers. Reproduced from [2]

The main benefit of activation functions is introducing nonlinearity into the computation, enabling the network to model complex relationships between inputs and outputs. The most commonly used activation function is rectified linear unit (ReLU)

$$ReLU(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

During training, the network is presented with a set of input-output pairs ($\mathbf{x}$, $\mathbf{t}$), and the weights and biases are adjusted to minimize the error between the network's predicted output ($\mathbf{y}$) and the true output ($\mathbf{t}$).

This is typically done using an optimization algorithm such as stochastic gradient descent (SGD), which iteratively updates the weights based on the gradient of the loss function $C(\mathbf{y}, \mathbf{t})$ with respect to the weights.

2.1 Backpropagation

In chapter 2, I mentioned that neural networks use an algorithm called stochastic gradient descent to update weights and biases based on the gradient of a loss function. However, I didn't mention how to compute this gradient. We do it by using an algorithm known as backpropagation. The main idea behind backpropagation is to calculate the partial derivative of the cost function with respect to weights ($\partial C/\partial w$) and biases ($\partial C/\partial b$) and adjust them in a way that minimizes the cost function [3].

As the name "backpropagation" suggests, we will first calculate the error in the output layer (equation (3)), and then backpropagate it through the whole network (equation (4)).

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} f'(z_j^L) \quad \Leftrightarrow \quad \delta^L = \left\langle \nabla_a C, f'(z^L) \right\rangle, \tag{3}$$

$$\delta^l = \left\langle \left(w^{l+1}\right)^T \delta^{l+1}, f'(z^l) \right\rangle, \tag{4}$$

we can think of applying the transpose of the weight matrix ($w^{(l+1)}$) as moving the error from the $(l+1)$-th layer to the l-th layer. With that, we can calculate the error δ^l for all the layers in the neural network. Now we are ready to write down equations that describe how the loss function changes with respect to biases (equation (5)) and weights (equation (6)).

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad \Leftrightarrow \quad \frac{\partial C}{\partial b} = \delta, \tag{5}$$

,

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l. \tag{6}$$

Now that we have gained knowledge of backpropagation, we are equipped to describe the complete neural network algorithm.

2.2 Neural netowork algorithm

Neural networks rely on following algorithm:

1. Randomly initialize weights and biases.
2. Set the activation of the input layer, $\mathbf{a}^1$, based on the input data $\mathbf{x}$.
3. Calculate the activation for each layer using the feedforward algorithm (equation (1)).
4. Compute the loss function.
5. Use the backpropagation algorithm (equations (3) to (6)) to calculate the gradient of the loss function.
6. Update the weights and biases using gradient descent

$$w^l \to w^l - \eta \sum_{\mathbf{x}} \delta^{\mathbf{x},l} \left(a^{\mathbf{x},l-1}\right)^T, \tag{7}$$

$$b^l \to b^l - \eta \sum_{\mathbf{x}} \delta^{\mathbf{x},l}, \tag{8}$$

where η is learning rate.

7. Repeat steps 2-6 with a different input-output pair.

3 Convolutional neural networks

ANNs have achieved success in solving a wide range of classification problems. However, traditional ANNs do not consider the spatial structure of input data, which may limit their ability to capture local patterns. To address this limitation, convolutional neural networks (CNNs) were developed and have become a powerful tool for tasks such as image and signal processing.

CNNs consist of a sequence of convolutional layers followed by pooling layers, typically concluding with at least one fully-connected layer for classification purposes (see Figure 3). As CNNs progress through their convolutional layers, they gain additional complexity. The initial layers are capable of recognizing edges, followed by recognition of simple shapes such as corners, squares, and circles, and so on. This progressive feature extraction allows CNNs to capture increasingly abstract and high-level representations of the input data.

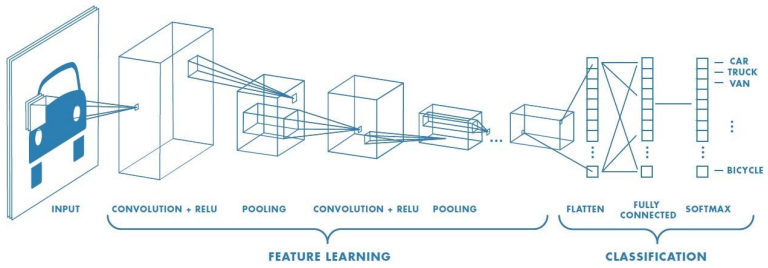

Figure 2: Architecture of Convolutional Neural Network. Reproduced from [4]

3.1 Convolutional layer

As the name suggests, this layer employs a mathematical operation called convolution,

$$s(t) = (w \star x)(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{9}$$

This operation is extensively described in [1]. In each convolutional layer, there are multiple feature detectors, also known as filters or weights (w). These filters traverse the input data, examining whether specific features are present. This evaluation is achieved by computing the dot product between the filter and the corresponding input (x). The process is repeated for every location in the input, resulting in a collection of feature maps (s). Each feature map represents the response of a particular filter at various locations in the input.

$$O_{(i,j),n} = f(\sum_{k=0}^{K-1}\sum_{l=0}^{K-1}\sum_{m=0}^{M-1} W_{(k,l),m,n} x_{(i+k,j+l),m}) \tag{10}$$

Where $O_{(i,j),n}$ represents the value of the n-th filter at position (i, j), f is the activation function, and $W_{(k,l),m,n}$ and $x_{(i-k,j-l),m}$ are the values of the filter and input at position $(k, l), m, n$ and $(i - k, j - l), m$, respectively.

Filters in CNNs are typically sized as 3×3 or 5×5. Similar to weights in ANNs, filters are not predetermined but learned through the backpropagation and gradient descent algorithm.

3.2 Pooling layer

Pooling layers often follow convolutional layers and they conduct dimensionality reduction, reducing the number of parameters for the next layer's input. A pooling layer also has a filter that sweeps through the input, but this filter does not have weights, which means it does not "learn". Instead, it replaces the output at a certain location with a summary statistic of nearby outputs. There are many different types of pooling, but two of the most popular ones are:

- Average pooling: The filter calculates the average value of nearby inputs.
- Max pooling: The filter chooses the maximum value of nearby inputs.

Someone might think that a lot of information is lost during pooling, but it improves computational efficiency and limits the risk of overfitting. Pooling also makes the representation of the input approximately invariant to small changes. This is helpful because most of the time we do not care about the exact position of a feature; we just need to know if a feature is present and its rough position. By summarizing nearby outputs, pooling helps capture the presence of important features while reducing the computational burden and enhancing generalization capabilities.

4 Complex-valued convolutional neural networks

Various image processing techniques use complex-valued representations of images. One notable example is the use of the Fourier Transform, which has been widely used in applications such as image compression and denoising. This has led to the idea of introducing complex-valued weights into CNNs.

Complex-valued CNNs have shown to produce better results than their real-valued counterparts (with a similar number of parameters) even on real-valued image classification tasks [5]. However, even bigger improvements are observed when dealing with complex-valued images, such as magnetic resonance imaging (MRI) reconstructions [6] and synthetic aperture radar [7].

To adapt traditional CNNs to CvCNNs, several modifications need to be made in the network architecture and operations. Lets begin by intorducing complex-valued convolution

$$\begin{aligned} s_r(t) &= (w_r \star x_r)(t) - (w_i \star x_i)(t) \\ s_i(t) &= (w_i \star x_r)(t) + (w_r \star x_i)(t) \end{aligned} \tag{11}$$

where the indices r and i refer to the real and imaginary components of the input x, weights w, and feature maps s. The most commonly used activation functions in CvCNN are complex versions of ReLU:

$$modReLU(z) = ReLU(|z| + b)e^{i\theta_z} \tag{12}$$

$$\mathbb{C}ReLU(z) = ReLU(Re\{z\}) + iReLU(Im\{z\}) \tag{13}$$

$$zRelu(z) = \begin{cases} z, & \text{if } \theta_z \in \left[0, \frac{\pi}{2}\right] \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where θ_z is a phase of complex number and b in $modReLU$ is trainable parameter. Slight modification are also made to pooling layer:

$$\mathbf{s}^l[j] = \max_n(\mathbf{x}^{l-1}[j]) \tag{15}$$

where j represents an input channel and $\max_n$ represents the maximum operation performed on an $n \times n$ square. We also need to define the complex max operation:

$$\max(x_1, x_2, \ldots, x_n) := \max(x_1^r, x_2^r, \ldots, x_n^r) + \imath \max(x_1^i, x_2^i, \ldots, x_n^i). \tag{16}$$

Additional adjustments have to be made to cost functions and backpropagation in order to effectively handle complex-valued inputs and preserve the complex information throughout the network. For more technical details, you can refer to [5].

All these adjustments allow CvCNNs to treat complex-valued inputs holistically, considering both amplitude and phase simultaneously (eq. (11)). This is in contrast to combining amplitude and phase separately and treating them as independent input channels in a traditional CNN. By preserving the complex-valued nature of the data, CvCNNs can capture the inherent relationships and interactions between the amplitude and phase, which are essential in many complex signal processing tasks.

5 Comparison of neural networks for image classification

In this section, we compare the performance of ANNs, CNNs,and CvCNNs on two (real-valued) image classification datasets. The first dataset is MNIST, which consists of 60,000 handwritten digits for training and an additional 10,000 for testing. Each image in MNIST is a grayscale image with a resolution of 28×28 pixels. The second dataset is CIFAR-10, which contains 50,000 RGB color images for training and 10,000 for testing. Each image in CIFAR-10 has a size of 32×32 pixels and belongs to one of ten classes, such as bird, car, airplane, or horse.

All neural networks were implemented using the PyTorch package. The ANN used for the MNIST dataset had an input layer with 784 neurons, two hidden layers with 128 and 32 neurons respectively, and an output layer with 10 neurons. This architecture resulted in a total of 104,938 trainable parameters. The CNN architecture for MNIST consisted of two convolutional layers with 60 and 30 5×5 filters, followed by a 2×2 pooling layer. The

data was then flattened into a 1D array and classified using two fully connected layers with 480 and 10 neurons respectively. The CNN had a total of 109,508 parameters. The CvCNN architecture for MNIST was similar to the CNN, but with half the number of filters due to the complex-valued nature of the weights. This resulted in a similar number of parameters compared to the CNN, with a total of 88,516.

For both datasets, all networks were trained using the Adam optimizer for 25 epochs, and the cost function used was cross-entropy loss. The performance of each network on both MNIST and CIFAR-10 datasets is presented in Table 1.

	ANN [%]	CNN [%]	CvCNN [%]
MNIST	97,5	**99,2**	99,1
CIFAR-10	52,8	70,8	**74,2**

Table 1: Clasification accuracy for ANN, CNN, CvCNN on MNIST and CIFAR-10 datasets

As shown in table 1, both CNN and CvCNN outperformed the ANN in both classification tasks. The CNN and CvCNN achieved similar accuracies in classifying MNIST digits, but the CvCNN performed significantly better in the more challenging CIFAR-10 dataset, surpassing the accuracy of the real-valued neural network by more than 3%.

6 Magnetic resonance imaging reconstruction

Magnetic resonance imaging (MRI) is a non-invasive medical imaging technique that enables the visualization of anatomical structures within the human body. It relies on the utilization of a strong magnetic field B_0, which is typically applied along the z-direction. When a patient is placed within the MRI scanner, the hydrogen nuclei (protons) in their body align themselves with this magnetic field. Radiofrequncy (RF) pulse then exictes protons (flips them into $x-y$ plane), this leads to them precess around z axis with Larmor frequency

$$\omega_0 = \gamma B_0 \tag{17}$$

here γ is the gyromagnetic ratio. If a gradient $\mathbf{G} = (G_x, G_y, G_z)$ is applied to the magnetic field, the frequency of protons changes.

$$\omega(\mathbf{r}) = \omega_0 + \gamma \mathbf{G} \cdot \mathbf{r}. \tag{18}$$

As the protons relax and return to their original alignment with the magnetic field, they emit a weak radiofrequency signal. This emitted signal is known as the free induction decay (FID). The FID signal is measured using specialized coils called receiver coils:

$$S(t, \mathbf{G}) = \iiint_V \rho(\mathbf{r}) e^{i\gamma \mathbf{G}\cdot\mathbf{r}t} d\mathbf{r}, \tag{19}$$

here, ρ represents proton density, and t represents time. To simplify the derivation, we introduce $\mathbf{k} = \gamma \mathbf{G} t$. Now, we can rewrite equation (19) as follows:

$$S(\mathbf{k}) = \iiint_V \rho(\mathbf{r}) e^{i\mathbf{k}\cdot\mathbf{r}} d\mathbf{r}, \tag{20}$$

here, one can recognize many similarities to the Fourier transformation equation. Now, we can perform an inverse Fourier transformation to obtain the proton density, which is of interest to us.

$$\rho(\mathbf{r}) = \frac{1}{(2\pi)^3} \iiint S(\mathbf{k}) e^{-i\mathbf{k}\cdot\mathbf{r}} d\mathbf{k}. \tag{21}$$

In order to obtain the complete proton density distribution, it is necessary to acquire data from many different points in k-space by varying the gradients systematically. However, this is a relatively slow process. To address this issue, researchers have explored methods for undersampling k-space to reduce scan times. Traditionally, undersampled images were reconstructed using parallel imaging and compressed sensing (CS) techniques. Since k-space is complex-valued one can see CvCNNs as an excelent candidate for reconstructing images form k-space. In [8], they compared CvCNNs with different activation functions to CNNs in terms of their ability to reconstruct MRI images (Table 2).

	Input image	CNN (ReLU)	CvCNN (CReLU)	(modReLU)	(zReLU)
SSIM	0.76 ± 0.08	0.88 ± 0.07	**0.90 ± 0.06**	0.87 ± 0.07	0.89 ± 0.06

Table 2: Structural similarity index measure (SSIM) for diffrent reconstruction methods.

They also compared the CNNs and CvCNNs to the traditional CS method, and the results are presented in Figure 3. The comparison clearly demonstrates that both neural networks outperformed the traditional approach in MRI reconstruction. Furthermore, the complex-valued neural network exhibited superior performance compared to the real-valued network, highlighting the advantage of incorporating complex-valued representations in handling MRI data.

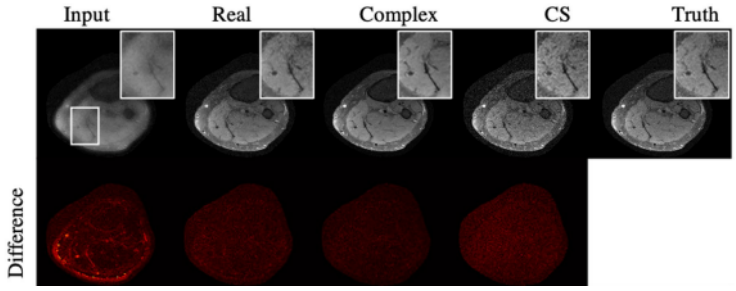

Figure 3: Comparison of different methods to reconstruct images from undersampled k-space. Reproduced from [8]

7 Conclusion

In conclusion, complex-valued convolutional neural networks offer a promising approach to image analysis by incorporating complex-valued representations. While traditional artificial

neural networks and convolutional neural networks have achieved remarkable results, CvCNNs aim to capture both the magnitude and phase information present in complex-valued data, enabling a more comprehensive understanding of images.

Additionally, we have also observed significant advancements in MRI reconstruction using CvCNNs, showcasing their potential in enhancing image reconstruction from under-sampled k-space.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] I. Neutelings, "Neural networks," Apr 2023. `https://tikz.net/neural_networks/`.

[3] M. A. Nielsen, *Neural networks and deep learning*. Determination Press, 2015.

[4] S. Saha, "A comprehensive guide to convolutional neural networks - the eli5 way," Dec 2018. `https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/`.

[5] C.-A. Popa, "Complex-valued convolutional neural networks for real-valued image classification," *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.

[6] E. Cole, J. Cheng, J. Pauly, and S. Vasanawala, "Analysis of deep complex-valued convolutional neural networks for mri reconstruction and phase-focused applications," *Magnetic Resonance in Medicine*, vol. 86, no. 2, p. 1093–1109, 2021.

[7] Z. Zhang, H. Wang, F. Xu, and Y.-Q. Jin, "Complex-valued convolutional neural network and its application in polarimetric sar image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 12, pp. 7177–7188, 2017.

[8] E. Cole, J. Cheng, J. Pauly, and S. Vasanawala, "Analysis of deep complex-valued convolutional neural networks for mri reconstruction and phase-focused applications," *Magnetic Resonance in Medicine*, vol. 86, no. 2, p. 1093–1109, 2021.