Jordan University of Science & Technology

Master's Programme in Radiologic Technology
Spring Semester 2023
Degree thesis

Attention Filter Gate U-Net: Learning from Frequency domain for Medical image Segmentation

Addressing problematic Feature Extraction from Spatial domain Through Fast Fourier Transformation Algorithm

Author: Ahmad Wajeeh Yousef E'layan

Author:	Ahmad Wajeeh Yousef E'layan,
Main Supervisor:	Research and Professor Radiologic Technology, Khalaf Abdelaziz Al Khalaf, Department of Allied Medical Sciences, Jordan University of Science and Technology (JUST)
Co-supervisor:	Research and teacher assistant Radiologic Technology, Haytham Al Ewaidat, Department of Allied Medical Sciences, Jordan University of Science and Technology (JUST)

1 Affirmation And Dedication

Dedication

Empty for now.

Affirmation

I hereby affirm that this Master thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification except as specified; nor has it been published.

_____

Student's name

City, date

Abstract

Background

Medical imaging diagnosis can be challenging due to low-resolution images caused by machine artifacts and patient movement. Researchers have explored algorithms and mathematical insights to enhance image quality and representation. One common task is segmentation, which requires the detection or localization of diseases around the tissue. New approaches and models using artificial intelligence, specifically computer vision, have been developed to improve the traditional methods that have not been entirely effective. Since the publication of the U-Net model paper, researchers have focused on building new model architectures to segment medical images more effectively. The Transformers model, a core technology behind many AI applications today, has been a game-changer. The Attention Gate was introduced in a paper and used with the U-Net model to increase performance. However, it did not solve certain computational cost issues and led researchers to investigate how to improve the Attention Gate in a different way while maintaining the same structure.

Aim

The aim was to improve the existing Attention Gate used in U-Net for medical image segmentation. The goal was to reduce the computational cost of training the model, improve feature extraction, and handle the problem of matrix multiplication used in CNN for feature extraction

Method

The Attention Filter Gate was developed to improve upon the Attention Gate. Instead of learning from the spatial domain, the model was converted to the frequency domain using the Fast Fourier Transformation (FFT). A weighted learnable matrix was used to filter features in the frequency domain, and FFT was implemented between up-sampling and down-sampling to reduce matrix multiplication. The method tackled computational cost, complexity algorithm, throughput, latency, FLOP, and enhanced feature extraction.

Results

Describe the main results of after finishing some Quantitative results empty for now

Conclusion

This thesis investigates the Attention Filter Gate to address problems such as computational cost and feature extraction, providing an alternative approach to medical image segmentation that is both efficient and effective. The method enhances feature extraction to reduce information loss between the encoder and decoder, and it provides a potential solution for throughput, latency, FLOP, and algorithm complexity issues. The Attention Filter Gate improves on the existing Attention Gate with intuitive tricks not addressed by previous methods.

Keywords:

Medical Segmentation, Neural networks, Transformers, U-Net model, Attention Gate , Fast Fourier Transformation (FFT),

Acknowledgement

I would like to express my sincere gratitude to Khalaf Abdelaziz Al Khalaf, my internal Co-supervisor Haytham Al Ewaidat, at Jordan University of Science and Technology, for their unwavering support, guidance, and valuable insights throughout the course of this thesis. Without their mentorship and encouragement, this research would not have been possible.

I am also grateful to Youness El Brag, my external supervisor at Abdelmalek Essaaˆdi University of Science and Technology, Faculty of Multi-Disciplinary Larache, Department of Computer Sciences, for his contributions and assistance that were integral to this study.

I would like to extend my appreciation to the examiner, [Examiner's Name], and the program director, [Program Director's Name], for their invaluable feedback and suggestions that helped to shape this research.

Lastly, I would like to express my deep gratitude to my parents for their unwavering support, encouragement, and belief in me throughout my academic and personal endeavors.

Contents

List of Abbreviations

BRB - Be Right Back
IMHO - In My Honest Opinion
L8r - Later
LOL - Laugh Out Loud
OMG - Oh My God!
ROFL - Rolling on the Floor Laughing

List of Figures

List of Tables

2 Introduction

2.1 Motivation

MRI has become the gold standard technique for precisely identifying patients' cardiac structures and etiology, guiding diagnostic and therapy decisions, due to its high picture quality, great soft-tissue contrast, and lack of ionizing radiation [2]. The left atrium (LA) is a critical component within the heart whose precise segmentation from medical images is critical for a variety of therapeutic applications, including cardiac illness diagnosis and therapy planning. The LA cavity has a relatively limited capacity, is confined by a thin atrial wall, and has a complicated structure [3]. Furthermore, the anatomical components around the atria have comparable intensities, which might confuse some segmentation algorithms [4].

As a result, analyzing atrial structures and determining and quantifying fibrosis distribution is difficult when utilizing manual atrial segmentation, which is a time-consuming, labor-intensive, and error-prone method [5]. Image segmentation of the left atrium (LA) can aid in overcoming the obstacles that stand in the way of accurate and effective diagnosis and assessment. LA segmentation measures atrial size and function, which are important imaging markers for several cardiovascular disorders including atrial fibrillation, stroke, and diastolic dysfunction. Currently, LA segmentation is done manually, which is time-consuming and observer-dependent as shown in the following Figure 2.1

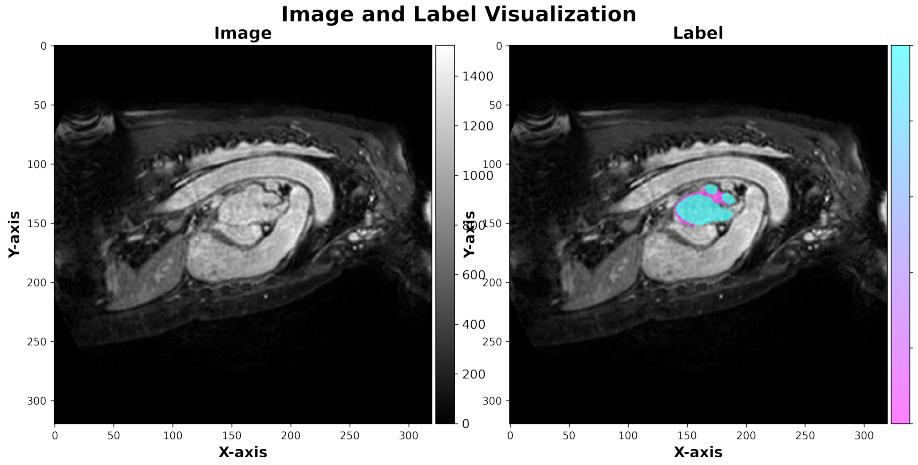


Figure 2.1: A Sample of the left atrium (LA) segmentation from DaTaset

Image segmentation is the process of dividing an image into multiple segments or regions that correspond to different objects or parts of the image, such as grayscale, color, spatial texture, and geometric shapes. The old and new methods of image

segmentation differ in terms of the techniques and algorithms used to perform this process [6].

Older methods of image segmentation typically relied on simple thresholding or clustering techniques, which worked by assigning pixels to different segments based on their intensity or color values. These methods could be effective in certain cases, but they were often limited by their inability to handle complex or noisy images [7, 8].

In recent years, new methods of image segmentation have emerged that rely on more advanced algorithms such as deep learning and computer vision techniques. However, the success rate was dependent on the availability of datasets for training. These methods are based on the use of neural networks and other machine learning techniques to identify patterns and features in images that can be used to segment them into different regions [9, 10].

One example of a new method of image segmentation is the use of convolutional neural networks (CNNs). These networks are trained on large datasets of labeled images, allowing them to learn to recognize different objects and features in images. Once trained, they can be used to segment new images by identifying the regions that correspond to different objects or parts of the image [11].

Another example of a new method of image segmentation is the use of superpixel segmentation. This technique works by grouping similar pixels together into larger regions or "superpixels" based on their color and texture features. This can be a more efficient way of segmenting images, as it reduces the number of individual pixels that need to be analyzed [11, 12].

Overall, while older methods of image segmentation can still be effective in certain cases, new methods based on deep learning and computer vision techniques offer more powerful and flexible tools for analyzing complex images, especially after owing hardware's considerable capabilities in image processing tasks [6, 8].

Left atrium (LA) segmentation is an important step in the analysis of cardiac magnetic resonance imaging (MRI) data for assessing the risk of atrial fibrillation and planning for ablation therapy [13, 14]. Many methods have been introduced for left atrium segmentation, like the Kalman filter, which shows promising results by handling the image noise to improve segmentation [15]. Noise can affect the quality of segmentation by blurring edges and boundaries between different regions of interest, introducing false contours.

The use of annotated 3D MRI data is essential for machine learning models and a crucial step for accurate network learning to extract special features from the image [16, 8]. Also, left atrium segmentation is a crucial goal after ablation therapy for those suffering from atrial fibrillation, with different LA shapes and sizes, and poor image quality, atrial segmentation can be challenging, especially using segmentation methods that can produce noise in images [15, 17].

In image segmentation, an attention filter is a technique used to improve the accuracy of the segmentation results by selectively focusing on specific regions of the image [18, 19]. The attention filter works by assigning different weights to the input image pixels based on their importance in the segmentation task. This weight assignment is typically learned by a neural network during the training phase [18]. The network learns to identify the relevant image features and assigns higher weights to those pixels that are more likely to be part of the object of interest [18]. The attention filter can be applied at various stages of the image segmentation pipeline,

such as during the feature extraction, pre-processing, or post-processing stages. In some cases, the attention filter is also used to filter out noise or unwanted image artifacts [19].

2.2 Problem description

The field of medical imaging faces challenges in accurately diagnosing and analyzing diseases due to various factors, including low-resolution images caused by machine artifacts and patient movement. One specific task in medical imaging is image segmentation, which involves the identification and localization of diseases or regions of interest within the tissue. Traditional segmentation methods have proven to be inadequate in achieving optimal results.

To address these limitations, researchers have turned to artificial intelligence techniques, particularly computer vision, to improve the effectiveness of medical image segmentation. The U-Net model, introduced in a seminal paper, has demonstrated significant advancements in this area. However, despite its success, the U-Net model still faces computational cost issues during training.

One technique used to enhance the U-Net model is the Attention Gate, which selectively focuses on relevant image features to improve segmentation performance. While the Attention Gate has shown promise, it has not fully resolved the computational cost challenges. Consequently, there is a need to explore alternative approaches to improve the Attention Gate's performance while maintaining its structure.

In this Thesis, we aim to develop a new approach called Filter Attention Gate we used within the U-Net model for medical image segmentation. Specifically, we seek to reduce the computational cost involved in training the model and enhance the feature extraction process, addressing the problem of matrix multiplication used in convolutional neural networks based on Fast Fourier Transformation. By tackling these challenges, we aim to provide an efficient and effective solution for medical image segmentation, ultimately improving diagnostic accuracy and patient care.

2.3 Aim

The aim of this Thesis is to develop a new mechanism called Filter Attention Gate within the U-Net model for medical image segmentation. The primary goal is to address the computational cost associated with training the model while improving feature extraction and overcoming the challenges posed by matrix multiplication in convolutional neural networks used for feature extraction. By achieving these objectives, this research aims to provide a simpler yet computationally efficient algorithm for medical image segmentation, ultimately enhancing the accuracy of disease detection and localization in medical imaging.

2.4 Objectives

The objectives of this Thesis are as follows:

1. Develop and improve Filter Attention Gate Based on FFT for medical image segmentation.

2. Reduce the computational cost associated with training the model.
3. Enhance feature extraction between the encoder and decoder.
4. Address the problem of matrix multiplication in CNN which Attention Gate used for feature extraction.
5. Provide Simple yet computationally efficient algorithm integrated within U-Net model

2.5 Study questions

In this thesis, we aim to address the following research questions:

1. How can the computational cost of training the U-Net model for medical image segmentation be reduced?
2. How does the proposed Filter Attention Gate mechanism compare to the existing Attention Gate in terms of computational efficiency and segmentation performance?
3. What impact does the integration of Fast Fourier Transformation (FFT) in the Filter Attention Gate have on the overall efficiency and effectiveness of medical image segmentation?

By addressing these research questions, this thesis aims to contribute to the field of medical image segmentation by developing a novel mechanism that reduces computational costs, improves feature extraction, and enhances the accuracy of disease detection and localization.

2.6 Delimitations

The delimitations of this Thesis are as follows:

- The proposed method will be evaluated on a specific dataset or datasets.
- The focus will be on improving the Filter Attention Gate within the U-Net model.
- The computational cost reduction will be assessed based on specific metrics.
- The feature extraction improvement will be measured through quantitative analysis.
- one common limitation comes with Hardware Resourecs that wil allows us yo run full the model Potienlial
- The thesis Study will not cover other aspects of medical image segmentation beyond the Attention Gate improvement.

3 Theory

3.1 Related Work

Biomedical 2D image segmentation has achieved remarkable accuracy with the use of Convolutional Neural Networks (CNNs), comparable to human performance [20]. Building on this success, researchers have explored the application of 3D CNNs for biomedical segmentation tasks [21, 22, 23].

The original U-Net architecture is designed for 2D image segmentation, and similar Dense Neural Network architectures have been developed for full heart segmentation [20]. To extend U-Net to 3D segmentation, the 3D U-Net was introduced by Cicek et al. [22]. One approach to process volumetric data is to take three perpendicular 2D slices as input and combine the multi-view information for 3D segmentation [24].

Another approach is to directly replace the 2D operations in the original U-Net with their 3D counterparts [20]. This 3D U-Net has been applied to multi-class whole heart segmentation using MR and CT data [21]. However, due to memory limitations of GPUs, these methods have either downsampled the input 3D data, resulting in reduced resolution, or predicted subvolumes of the data [25, 26, 27].

Some methods adopt a two-stage approach, where regions of interest (ROI) are first extracted using a localization network and then applied to the segmentation U-Net [28]. This method utilizes two stages of 3D U-Nets.

In our work, we focus on utilizing the 3D U-Net architecture to segment 3D volumes of the whole heart obtained from 4D flow MRI, which have been previously segmented using multi-atlas-based methods [29, 28, 21].

Automated medical image segmentation has gained significant attention in the image analysis community due to the labor-intensive and error-prone nature of manual labeling [30, 31]. CNNs, particularly U-Net, have shown promising results in tasks such as cardiac MR segmentation [30] and cancerous lung nodule detection [25], approaching near-radiologist level performance.

However, when target organs exhibit large inter-patient variation in shape and size, fully convolutional networks (FCNs) and U-Net often rely on multi-stage cascaded CNNs [21, 25, 10, 24, 27, 28]. This cascade approach extracts ROIs and performs dense predictions on the selected regions. Nonetheless, this strategy leads to redundant resource usage and model parameters [31, 24, 27, 28]. To address this issue, attention gates (AGs) have been proposed as a simple yet effective solution [21, 24, 27, 28]. CNN models equipped with AGs can be trained from scratch similar to FCN models, and the AGs learn to focus on target regions automatically.

In our method, we propose a novel mechanism called the Filter Attention Gate, integrated within the U-Net model, for medical image segmentation. The main

objective is to address the computational cost associated with model training, improve feature extraction, and overcome the challenges posed by matrix multiplication in convolutional neural networks. By achieving these goals, our research aims to provide a computationally efficient algorithm for medical image segmentation, ultimately enhancing the accuracy of disease detection and localization in medical imaging.

3.1.1 Semantic Segmentation

Semantic segmentation is the task to split a digital image into multiple segments where each segment represents a certain object. For semantic segmentation, multiple objects of the same class are not distinguished differently, which is not the case for instance segmentation. This report is focused on semantic segmentation. This section will discuss different methods used for Semantic segmentation starting with classical methods then moving to deep learning models.

3.1.2 Classical Methods

1. Manual Thresholding
This method depends mainly on a threshold value that splits the image to a binary image where each pixel is either zero or one. Multiple thresholds are needed when the number of classes is larger than two. This method needs expert interaction to define those thresholds and it also lacks generality as it is impossible to define a threshold for each class. **2. Clustering Methods**
Clustering is to group similar pixels into the same cluster/label. There are different algorithms for clustering. However, the most known algorithm is called K-means where 'K' represents the number of clusters in the image. The algorithm is based on the iterative method as it assumes k centroids at the beginning of the k clusters. Then, it tries to find pixels that belong to each cluster by calculating the distance between all the pixels and the centroids of the clusters. The distance function is the difference between the pixel values. When two pixels have similar grayscale values, the distance between them will be small. This indicates that these two pixels should be clustered into the same cluster. On the other hand, when the distance is large, it indicates that these two pixels belong to different clusters. Finally, it calculates new k centroids according to the pixels in each cluster. The cluster centroid is the average grayscale value of the pixels in this cluster. The loop goes over and over until it saturates when centroids do not change. This method also lacks generality as it needs variable k to be defined for each image. Moreover, it might produce the wrong results because of bad initialization.

3. Histogram-Based Methods
A histogram is built using all pixels of the image, then peaks and valleys make it easier to distinguish different objects. The algorithm could be applied to multiple images at the same time in the same way. However, it becomes harder to split the image to classes when there are no peaks or valleys or they are in a small range. **4. Edge Detection**
Edge detection techniques are used to identify edges in the image. These techniques are built on heuristics that look for discontinuity in the image. Although the result is groups of disconnected edges, with some heuristics they can form objects and

shapes. So, these techniques are used as a base for segmentation. This method depends on heuristics which makes it not viable for general cases.

3.2 Feed Forward Neural Network

In this section, we will delve into the topic of single-layer neural networks, which includes some of the classical approaches to neural computing and learning problems. We will first discuss the representational power of single-layer networks and their learning algorithms, providing examples of their usage. Subsequently, we will address the representational limitations encountered by single-layer networks.

Two classical models will be described in the first part of this chapter: the Perceptron, proposed by Rosenblatt in the late 1950s [32] An artificial neural network is an application that is non-linear with respect to its parameters θ, and it associates an input x with an output $y = f(x, \theta)$. For simplicity, we assume that y is unidimensional, although it could also be multi-dimensional. The function f has a specific form that will be explained. Neural networks can be used for regression or classification tasks. The parameters θ are estimated from a learning sample, and the optimization function is non-convex, which can result in finding local minimizers. The success of neural networks stems from the universal approximation theorem proposed by Cybenko (1989) and Hornik (1991) [32, 33]. Furthermore, LeCun (1986) introduced an efficient method called backpropagation of the gradient, which enables the computation of gradients for neural networks and facilitates the attainment of local minimizers for quadratic criteria [34]. The use of neural networks, with their flexibility in representation and their efficient optimization techniques, has contributed significantly to the advancement of artificial intelligence and machine learning.

3.2.1 Artificial Neuron

An artificial neuron, denoted as f_j, is a function of the input $x = (x_1, \ldots, x_d)$ weighted by a vector of connection weights $w_j = (w_{j,1}, \ldots, w_{j,d})$, along with a neuron bias b_j. This neuron is associated with an activation function φ, and its output y_j is computed as follows:

$$y_j = f_j(x) = \varphi\left(\sum_{i=1}^{d} w_{j,i}x_i + b_j\right). \tag{3.1}$$

Various activation functions can be considered for the artificial neuron. Some commonly used activation functions include:

- The identity function:
$$\varphi(x) = x \tag{3.2}$$
- The sigmoid function (or logistic function):
$$\varphi(x) = \frac{1}{1 + \exp(-x)} \tag{3.3}$$

- The hyperbolic tangent function :

$$\tanh : \varphi(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1} \tag{3.4}$$

- The hard threshold function:

$$\varphi_\beta(x) = \begin{cases} 1, & \text{if } x \geq \beta \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

- The Rectified Linear Unit (ReLU) activation function:

$$\varphi(x) = \max(0, x) \tag{3.6}$$

The artificial neuron can be represented schematically as follows, where $\Sigma = \langle w_j, x \rangle + b_j$:

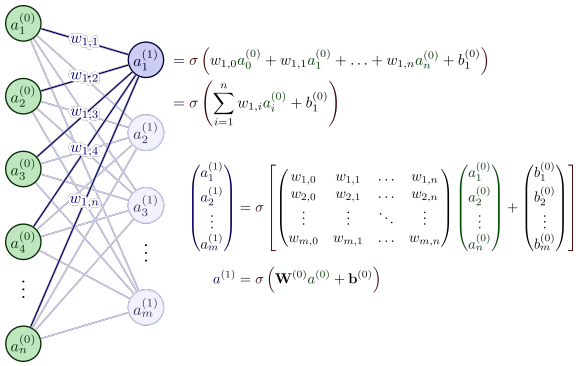


Figure 3.1: Schematic representation of an artificial neuron.

These activation functions introduce non-linearity into the neural network, enabling it to learn complex patterns and make non-linear decisions. The choice of activation function depends on the nature of the problem and the desired behavior of the neural network.

3.2.2 Activation and output rules

after we introduce a general Overview of Neural networks we need to understand the connectivity between neurons which leads us to activation functions [35] [33], moreover, We also need a rule which gives the effect of the total input on the activation of the unit. We need a function F_k which takes the total input $s_k^{(t)}$ and the current activation $y_k^{(t)}$ and produces a new value of the activation of the unit k:

$$y_k^{(t+1)} = F_k(y_k^{(t)}, s_k^{(t)}) \tag{3.7}$$

Often, the activation function is a non-decreasing function of the total input of the unit:

$$y_k^{(t+1)} = F_k(s_k^{(t)}) \tag{3.8}$$

In some cases, the output of a unit can be a stochastic function of the total input of the unit. In that case, the activation is not deterministically determined by the

neuron input, but the neuron input determines the probability p that a neuron gets a high activation value:

$$p(y_k = 1|s_k) = \sigma(s_k) \tag{3.9}$$

Historically, the sigmoid function was mostly used as the activation function since it is differentiable and allows us to keep values in the interval [0, 1]. Nevertheless, it has a problem: its gradient is very close to 0 when $|x|$ is not close to 0. Figure 3 represents the sigmoid function and its derivative. With neural networks with a high number of layers (which is the case for deep learning), this causes trouble for the backpropagation algorithm to estimate the parameters (backpropagation is explained in the following). This is why the sigmoid function was supplanted by the rectified linear function. This function is not differentiable at 0, but in practice, this is not really a problem since the probability to have an entry equal to 0 is generally null. The ReLU function also has a sparsification effect. The ReLU function and its derivative are equal to 0 for negative values, and no information can be obtained in this case for such a unit. This is why it is advised to add a small positive bias to ensure that each unit is active. Several variations of the ReLU function are considered to make sure that all units have a non-vanishing gradient [36] and that for $x < 0$, the derivative is not equal to 0. Namely

$$\varphi(x) = \max(x, 0) + \alpha \min(x, 0) \tag{3.10}$$

where α is either a fixed parameter set to a small positive value, or a parameter to estimate.

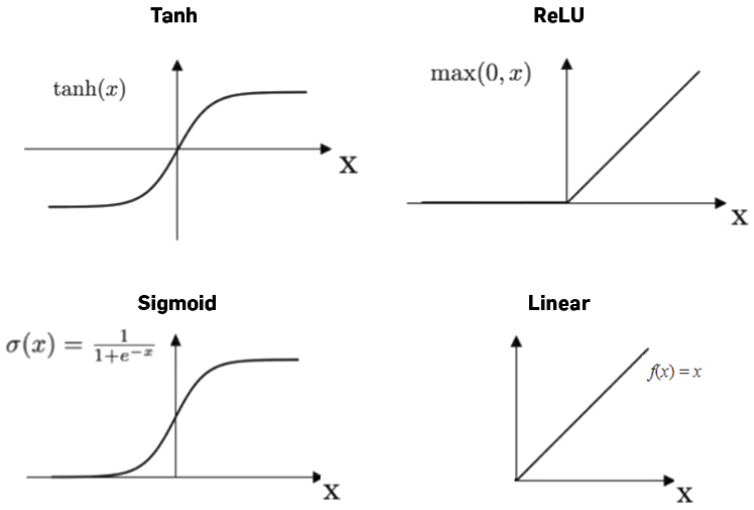


Figure 3.2: Various activation functions for a unit

3.3 Multi-Layer Perceptron

A multi-layer perceptron [37] (or neural network) is a structure composed of several hidden layers of neurons, where the output of a neuron in one layer becomes the input of a neuron in the next layer. Additionally, the output of a neuron can also be the input of a neuron in the same layer or in previous layers (this is the case

for recurrent neural networks). The last layer called the output layer, may apply a different activation function compared to the hidden layers, depending on the type of problem at hand, such as regression or classification. Figure 4 represents a neural network with three input variables, one output variable, and two hidden layers.

Multi-layer perceptrons shown in Fig Figure.3.3 have a basic architecture where each unit (or neuron) in a layer is connected to all the units in the next layer but has no direct connection with the neurons in the same layer. The architectural parameters include the number of hidden layers and the number of neurons in each layer. The choice of activation functions is also left to the user. For the output layer, as mentioned earlier, the activation function is generally different from the ones used in the hidden layers.

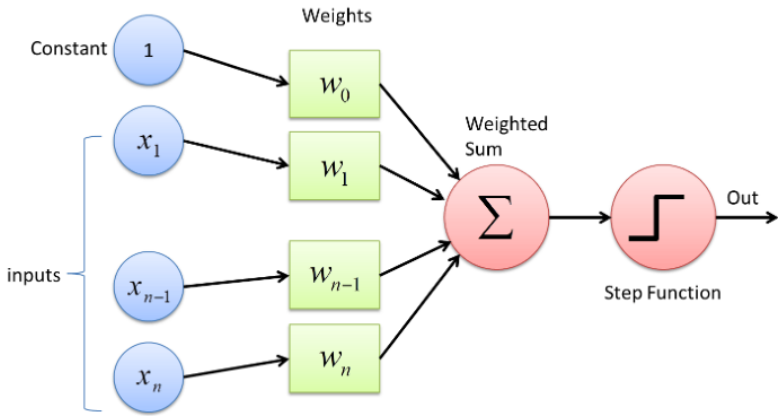


Figure 3.3: basic neural network multi-layer perceptron

In regression tasks, no activation function is applied to the output layer. For binary classification, where the output provides a prediction of $P(Y = 1/X)$ (since this value is in the range $[0, 1]$), the sigmoid activation function is commonly used. In multi-class classification, the output layer contains one neuron per class i, giving a prediction of $P(Y = i/X)$. The sum of these predicted probabilities should be equal to 1. The multidimensional softmax function is typically employed for this purpose:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Let us summarize the mathematical formulation of a multi-layer perceptron with L hidden layers. We set $h^{(0)}(x) = x$.

For $k = 1, \ldots, L$ (hidden layers):

$$a^{(k)}(x) = b^{(k)} + W^{(k)} h^{(k-1)}(x)$$
$$h^{(k)}(x) = \varphi(a^{(k)}(x))$$

For $k = L + 1$ (output layer):

$$a^{(L+1)}(x) = b^{(L+1)} + W^{(L+1)} h^{(L)}(x)$$
$$h^{(L+1)}(x) = \psi(a^{(L+1)}(x)) := f(x, \theta)$$

where φ is the activation function and ψ is the output layer activation function (e.g., softmax for multiclass classification). At each step, $W^{(k)}$ is a matrix with

the number of rows equal to the number of neurons in layer k and the number of columns equal to the number of neurons in layer $k-1$.

In the subsequent step, we define a loss function and apply backpropagation to update the weights. The error in the output of the neural network is defined by the loss function. The loss function used depends on the application, for instance, a classification, regression, or in our case, segmentation problem. An assumption can be made that the error is normally distributed about the prediction y. Hence, the Maximum Likelihood Estimate (MLE) for the normal distribution can be optimized by maximizing the Mean Squared Error (MSE), where y denotes the true value and $\hat{y}$ denotes the estimated value.

$$e(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2$$

The gradient can now be computed with respect to the corresponding weights and updated. The learning rate α decides the magnitude of change in the weights after each iteration or epoch, i.e., one pass of the neural network training over the training dataset. For the purpose of the application, more complex optimizers along with momentum are used for the training to converge faster to a solution. The equations and process detailing the weight update mechanism are referred to in [21]. One gradient update can be shown in the gradient of the error function with respect to its weights:

$$\nabla e = \left(\frac{\partial e}{\partial w_n}, \frac{\partial e}{\partial w_{n-1}}, \ldots, \frac{\partial e}{\partial w_1}\right)$$

The chain rule of partial derivatives can be used to compute the gradient. For w_2, the gradient is:

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h} \cdot \frac{\partial h}{\partial w_2}$$

Hence, the weight can be updated according to the equation:

$$w_2' = w_2 - \alpha \odot \frac{\partial e}{\partial w_2}$$

where $\odot$ denotes the element-wise product and α is the learning rate that decides the magnitude of change of the weight update. Sometimes, updates to the structure of the neural network may be required, which changes the analytical solution for gradient updates. The application of numerical optimization techniques can allow for such a change without requiring the derivation of the analytical solution for such a gradient update. Deep neural networks are susceptible to the vanishing gradient/exploding gradient problem [33], which is the reason for the application of more complex types of neural networks.

3.4 Convolutional Neural Networks

Inputs to the neural network may have features that are highly correlated to the features adjacent to them. This is generally observed in image classification and segmentation problems. Convolutional Neural Networks (CNNs) can be used for

extracting information in such cases where feature extraction from images is required for later tasks such as regression, classification, and segmentation. In theory, a conventional feed-forward neural network is also capable of image processing, with the drawback of a higher number of parameters required to be learned as compared to a CNN. Instead of using a weight for each pixel of the image, a CNN uses a *filter* over the *inputs* i.e. a sliding window with a certain stride over the pixel intensities to create an intermediary output that is a function of the input and the filter, this process termed *convolution* Figure.3.4.

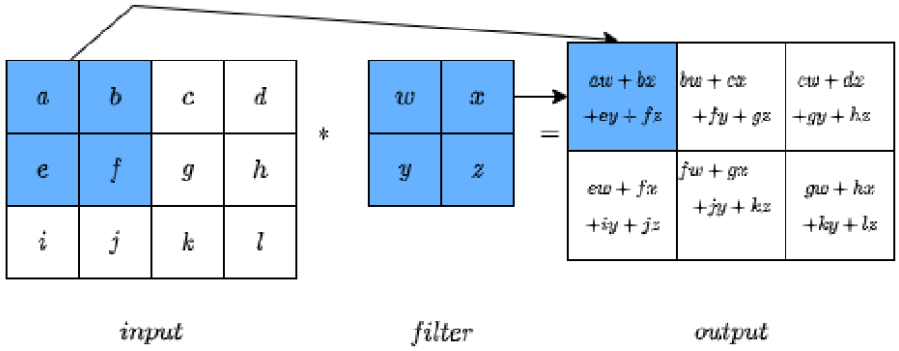


Figure 3.4: Illustrative 2D convolution: The dot product of *input* and *filter* results in the *output*

The intensity/brightness of each pixel of the image (voxel in 3D case) is generally rescaled and mapped to a range such as (0, 1). This can be seen in Figure.3.4 as an example. It is noted that zero-padding may be performed on the input to allow for convolution involving all the input pixel or voxel intensities, i.e., 0-intensity pixels are added around all the edges and corners. Other forms of padding are also adopted depending on the application. During each convolution step as described above, a product of the input and filter is computed. In the case of Figure.3.4 with 2D inputs or pixels, the input shape is 1×4 and 4×1, which results in an output of shape 1×1. The stride of 1 moves the sliding window of 2×2 to the next position with the output being computed on the right-hand side. Increasing the strides will result in a smaller output size; it can be noted that the size of the output is always smaller than the input size.

The convolution operation in the 2D space can be defined as follows [38]:

$$y_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} x_{1+as,j+bs} F_{a,b}$$

where x denotes the input at indices i and j, F denotes the filter, m denotes the size of the filter, and s is the stride.

For the purpose of extracting features from images, edges are of interest. Therefore, a difference between adjacent pixels/voxels can determine the edge, as in the case of foreground vs background. Size reduction of the input is also a concern for faster image processing and can be done by a method called pooling. An illustrative example of max-pooling can be seen in Figure.3.5.

The concept of pooling can also be described as a sliding window of a mathematical operation, such as the maximum value (max), applied in each case. Generally, no

overlap is considered between the selected regions. This can result in the edge cases being overlooked, but padding may also be used. Settings related to pooling and filter size are both arbitrarily decided and hence considered to be hyperparameters of the neural network.

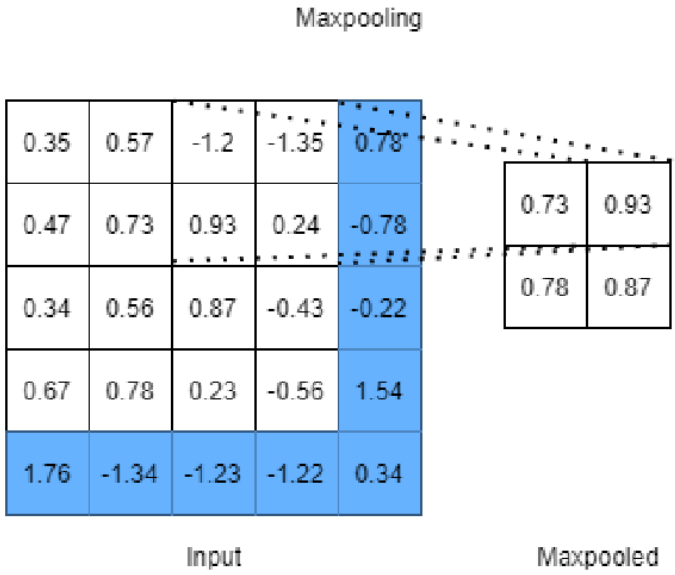


Figure 3.5: Maxpooling is being performed on the input by a window of size 2 ˆ 2. The cases with *edges* are either padded with zeros or just ignored

The filter shown in Figure.3.4 is a high-pass filter. A convolutional neural network (CNN) would be of limited use if it could only learn with one filter, meaning it would be able to extract only one type of feature from the input. For instance, an *edge* detection filter works where the intensity changes drastically (e.g., from black to white). Hence, multiple independently operating filters are used for training. In image processing, one filter is used for each of the RGB channels if available; otherwise, the number of filters is decided by the modeler. The construction of a CNN involves multiple layers of filtering and pooling, which provide a feature set to the network resembling a feed-forward network for the purpose of the end goal task, such as classification, regression, or segmentation.

3.5 Upsampling

For a neural network to generate images or image maps such as in the case of semantic segmentation, it generally involves the application of upsampling from a low resolution to a higher resolution. There are many different methods to perform an upsampling operation such as nearest neighbor interpolation, linear interpolation, bilinear interpolation, trilinear interpolation, bicubic interpolation, and various others found in literature [35]. Let's go backward in Figure.3.4 from the section 3.3 , if we want to associate 1 value to 9 other values in a matrix, it will be termed a one-to-many relationship. This is similar to going backward in a convolution operation Figure.3.6

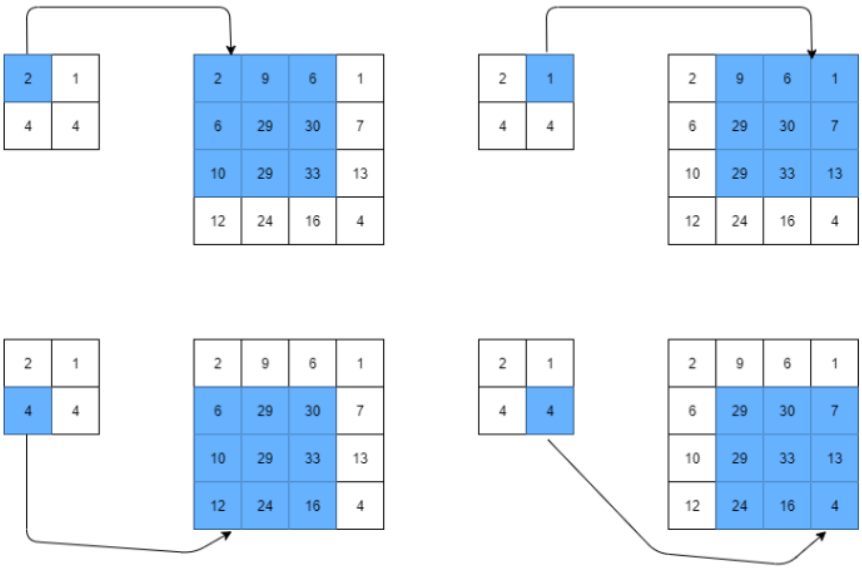


Figure 3.6: Illustrative Upsampling - Going backward of a convolution

3.5.1 Convolution operation

When a convolution operation is applied on a $4x4 matrix$ using a 3x3 kernel without padding and a stride of 1, the output will be a $2x2 matrix$. The operation in Figure.3.7 computes the sum of element-wise matrix multiplication between the input and kernel. This can be done only 4 times in the illustrative example

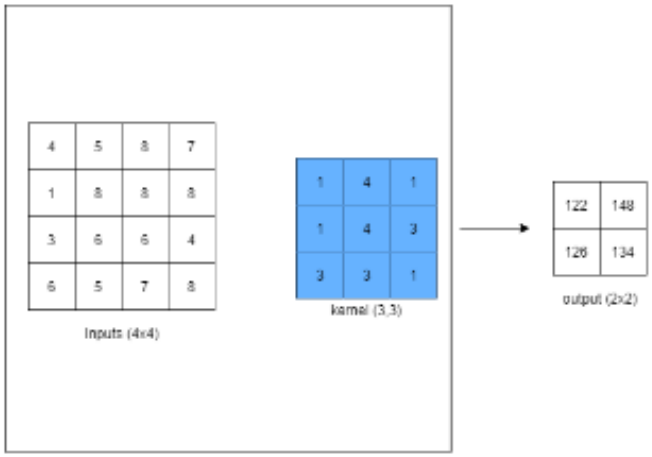


Figure 3.7: Illustrative Convolution operation

3.5.2 Transposed Convolution

A convolution operation can be represented as a kernel matrix that is arranged in the form of matrix multiplication to perform convolution operations. Shown in the Figure.3.8

Figure 3.8: Illustrative Convolution operation

This kernel can be arranged as in Figure.3.9

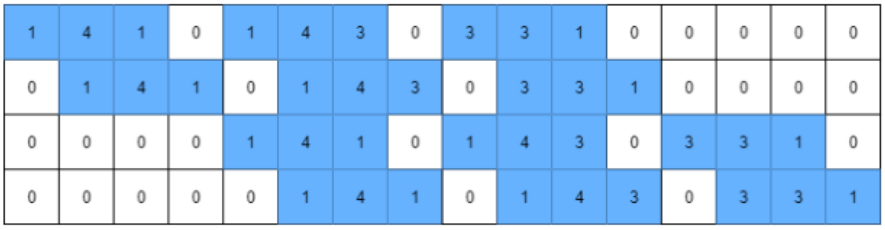


Figure 3.9: Illustrative Convolution matrix (4,16)

3.6 Non-parametric Activation Functions

Complex-valued neural networks (CVNNs) [39] are a powerful modeling tool for domains where data can be naturally interpreted in terms of complex numbers. However, several analytical properties of the complex domain (e.g., holomorphicity) make the design of CVNNs a more challenging task than their real counterpart. In this work, we consider the problem of flexible activation functions (AFs) in the complex domain, i.e., AFs endowed with sufficient degrees of freedom to adapt their shape given the training data. While this problem has received considerable attention in the real case, a very limited literature exists for CVNNs, where most activation functions are generally developed in a split fashion (i.e., by considering the real and imaginary parts of the activation separately) or with simple phase-amplitude techniques. In our work, we propose the use of our own Attention Filter Gate, which combines both Softmax and Sigmoid activation functions to handle frequencies in the Frequency Domain before taking the inverse of FFT. We will provide a detailed explanation of this method in Section **??**.

3.6.1 Sigmoid Activation Function

As stated in the section, choosing a proper activation function in Equation 3.11 is more challenging than in the real case due to Liouville's theorem, which states that the only complex-valued functions that are bounded and analytic everywhere are constants. Therefore, in practice, one needs to choose between boundedness and analyticity. Before the introduction of the sigmoid activation [40], most activation functions in the real case were bounded. As a result, initial approaches to design

CVNNs often preferred non-analytic functions to preserve boundedness. One common approach was to apply real-valued activation functions separately to the real and imaginary parts [41].

$$g(z) = g_R(\mathrm{Re}(z)) + ig_R(\mathrm{Im}(z)) \tag{3.11}$$

Here, z represents a generic input to the activation function in Equation 3.11, and $g_R(\cdot)$ denotes some real-valued activation function, such as the sigmoid (Equation 3.14). This approach is known as a split activation function. An example of the split-tanh activation function, showing the magnitude and phase variations when changing the activation, is illustrated in Figure 3.10. Early proponents of this approach can be found in [41] and [42].

Another class of non-analytic activation functions commonly used is the phase-amplitude (PA) functions popularized by [43, 44]:

$$g(z) = \frac{z}{c + \frac{|z|}{r}} \tag{3.12}$$

$$g(z) = \tanh\left(\frac{|z|}{m}\right) \exp\{i\varphi(z)\} \tag{3.13}$$

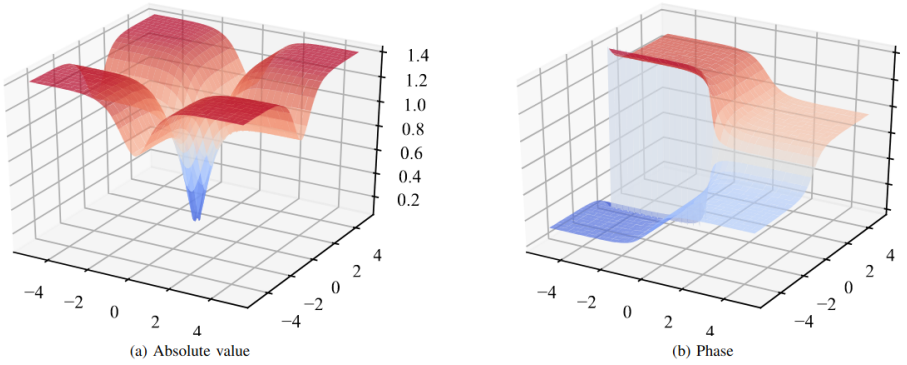


Figure 3.10: Example of split activation function with $SigmoidComplex(\cdot)$ processing both the real and imaginary parts of the input. (a) Magnitude of the output. (b) Phase of the output.

Here, $\varphi(z)$ represents the phase of z, while c, r, and m are positive constants typically set to 1. PA functions can be seen as a natural generalization of real-valued squashing functions, such as the sigmoid, as they maintain a bounded magnitude while preserving the phase of z.

1. Code for the SigmoidComplex used in our method:

   The following code snippet implements the SigmoidComplex function in Python:

```
class SigmoidComplex(nn.Module):
    def __init__(self):
        super(SigmoidComplex, self).__init__()

    def forward(self, x):
        if x.dtype == torch.complex64:
```

```
            # Calculate the magnitude of complex numbers
            magnitude = torch.abs(x)

            # Apply sigmoid activation function to the magnitude
            sigmoid_magnitude = torch.sigmoid(magnitude)

            # Normalize the complex numbers using the sigmoid
    magnitude and the original phase
            norm_real = sigmoid_magnitude * torch.cos(torch.angle(x))
            norm_imag = sigmoid_magnitude * torch.sin(torch.angle(x))
            normalized_complex = torch.complex(norm_real, norm_imag)

            return normalized_complex

        return torch.sigmoid(x)
```

The SigmoidComplex function is utilized in our method to process complex inputs. It incorporates the concept of complex-valued neural networks (CVNNs) and is designed to apply a sigmoid activation function to complex numbers. It follows the theorem of CVNNs, which states that the SigmoidComplex activation should be computed as follows:

2. Formalization of SigmoidComplex based on the theorem of CVNNs:

   The SigmoidComplex function is defined as follows:

$$\text{SigmoidComplex}(x) = \begin{cases} \sigma(|x|) \cdot \cos(\text{angle}(x)), & \text{if } x \text{ is complex,} \\ \sigma(x), & \text{otherwise.} \end{cases} \tag{3.14}$$

   In this equation, $\sigma(\cdot)$ represents the sigmoid function, $|x|$ denotes the magnitude of the complex number x, and $\text{angle}(x)$ represents the angle or phase of x. The SigmoidComplex activation function takes a complex input x and applies the sigmoid function to the magnitude while preserving the phase if x is complex. If x is not complex (i.e., a real number), the sigmoid function is applied directly.

   The SigmoidComplex function is a fundamental component in our method for handling complex-valued data within neural networks, providing a non-linear activation that is well-suited for complex inputs.

3.6.2 Softmax Activation Function

In this subsection, we compare different approaches for designing neural networks with softmax activation functions in their output layer. Specifically, we compare a real-valued neural network that takes the real and imaginary components of the coefficients as separate inputs, a complex-valued neural network (CVNN) with modReLU activation functions, and a CVNN employing the proposed split-KAF (Kernel Activation Function).

For the CVNNs, we utilize a variation of the softmax function to handle complex-valued activations, denoted as $\text{softmaxn}(h)$, where $h \in \mathbb{C}$ represents the complex-valued activations and $C = 1$ is the number of classes for our problem. The $\text{softmaxn}(h)$ function is defined as follows:

$$\text{softmaxn}(h) = \frac{\exp\left\{\sum_{t=1}^{C}(\langle h_n\rangle^2 + \|h_n\|^2)\right\}}{\sum_{t=1}^{C}\exp\left\{(\langle h_t\rangle^2 + \|h_t\|^2)\right\}}, \tag{3.15}$$

where $\langle h_n\rangle$ denotes the real part of the complex activation h_n, and $\|h_n\|$ represents the imaginary part of h_n.

To train the networks, we minimize the BCE-DICE Loss cross-entropy formulation using the same optimizer. The BCE-DICE Loss is commonly used for segmentation tasks and is adapted here to our problem.

Please note that the specific details of the optimizer used and the BCE-DICE Loss formulation can be found in the respective references or documentation of the methodology being employed.
Through our experiments, we observe that working in the complex domain and training our Attention Filter Gate (AFG) using the proposed split-KAF. This approach leverages the benefits of complex-valued neural networks, allowing for more expressive and accurate representations of complex data in the Frequency domain.

3.7 U-Net

The U-Net architecture, introduced by Ronneberger et al [10], is a convolutional neural network commonly used for image segmentation tasks. It has been widely adopted in various medical image analysis applications. U-Net addresses the challenge of training deep networks with a limited number of annotated samples. It consists of a contracting path and a symmetric expanding path, enabling precise localization and outperforming previous methods.

The U-Net architecture as shown in Figure.3.12 is built upon the fully convolutional network presented in Long work. A significant modification is the use of a large number of feature channels in the upsampling part. This allows the network to propagate context information to higher-resolution layers. The contracting side of the U-Net is symmetric to the expansive side, resulting in the characteristic U-shaped architecture. Unlike traditional architectures, U-Net does not include fully connected layers. Instead, it utilizes the valid parts of each convolution, ensuring that the segmentation map contains only pixels for which full context is available in the input image. The U-Net architecture has demonstrated excellent performance in various segmentation tasks, particularly in medical imaging. Its ability to capture both local and global information, along with its symmetric design, enables accurate segmentation and precise localization of structures within the images.

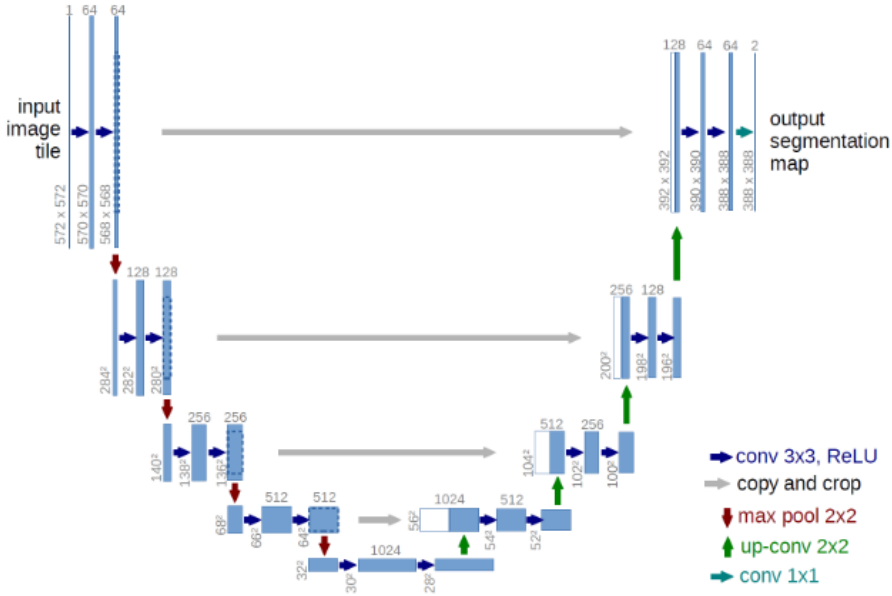


Figure 3.11: : U-Net architecture (illustrative for 32x32 pixels at lowest resolution). The blue boxes represent a multi-channel feature map with the number of channels present on top of each box. The size of the xy dimensions is at the bottom left of the box's edge. Copied feature maps are in white boxes. Each arrow denotes different operations

3.7.1 Tow Dimensional U-Net

A network for efficient Volumetric segmentation was first presented by Cicek et. al.[**?**]. We use the fully automated setup as referenced in the paper where we assume that a sparsely annotated training set exists. Trained on this dataset, the network segments new volumetric images. This network extends the U-Net as given by Ronneberger et. al. by replacing 2D operations with the corresponding 3D operations. Volumetric data is available in abundance in the biomedical industry. Annotation of such data with segmentation labels comes with its cons, since only 2D slices are shown, annotation of large volumes slice by slice is inefficient. Neighboring slices mostly show the same information.

The first U-Net proposed is designed as a 2D architecture [10], while the network proposed by Cicek et. al.[45] takes 3D volumes for input to process them with corresponding 3D operations such as 3D convolutions, 3D max-pooling, and also 3D upsampling layers. Deep Convolutional Neural Networks (CNNs) are the pinnacle in performance for multi-class segmentation of medical images. However, the most common problem when dealing

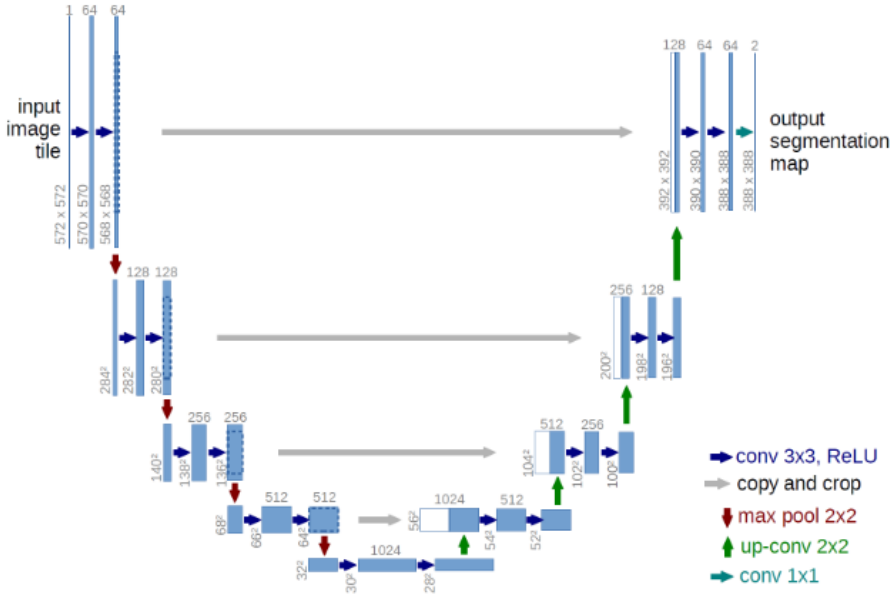


Figure 3.12: U-Net architecture (illustrative for 32x32 pixels at lowest resolution). The blue boxes represent a multi-channel feature map with the number of channels present on top of each box. The size of the xy dimensions is at the bottom left of the box's edge. Copied feature maps are in white boxes. Each arrow denotes different operations

with high resolution and large 3D data, the volumes as input into the deep CNNs have to be either cropped or downsampled due to limitations in memory and computation. The above operations lead to loss of resolution and class imbalance in the input data batches, thus downgrading the performances of segmentation algorithms. Wang et. al. propose a two-stage modified U-Net framework applied to a variety of multi-modal 3D cardiac images for full cardiac segmentation. Other efforts such as introducing several auxiliary loss functions, anatomically constrained CNNs where some nodes are constrained and shape priors are used, and Hierarchical 3D fully convolutional networks for multi-organ segmentation. A 3D V-Net has also been proposed where the network architecture is more like a V shape since some convolution steps, copy and crop steps are skipped from the U-Net which can make it faster. A novel cost function is used to optimize the hyperparameters

3.7.2 Uncertainty Estimation using a Bayesian 3D U-Net

While the sigmoid output or the last layer output of a U-Net can provide a measure of uncertainty, it cannot be directly used as an accurate estimate. This is because the sigmoid output values are dependent on the inferred samples being subjectively close to the training distribution. Additionally, the activation function used in the last layer squeezes the output, further limiting its suitability as an uncertainty estimate, especially for samples that are far from the training distribution, such as deformed or diseased parts of the heart. To address this limitation, Labonte et al. outlined a method for performing variational inference in their work [46]. This approach enables the estimation of uncertainty in Bayesian 3D U-Nets. Uncertainty estimation is crucial for interpretability and validation purposes, as it provides insights into the reliability and confidence of the segmentation results. The significance of uncertainty

estimation is illustrated in Figure 3.12. Traditional deep neural networks lack the capability to measure uncertainty, and their performance is highly dependent on the similarity between the test set and the training dataset. When presented with a test example that differs significantly from the training data, the network may produce incorrect segmentation outputs due to the lack of uncertainty estimation.

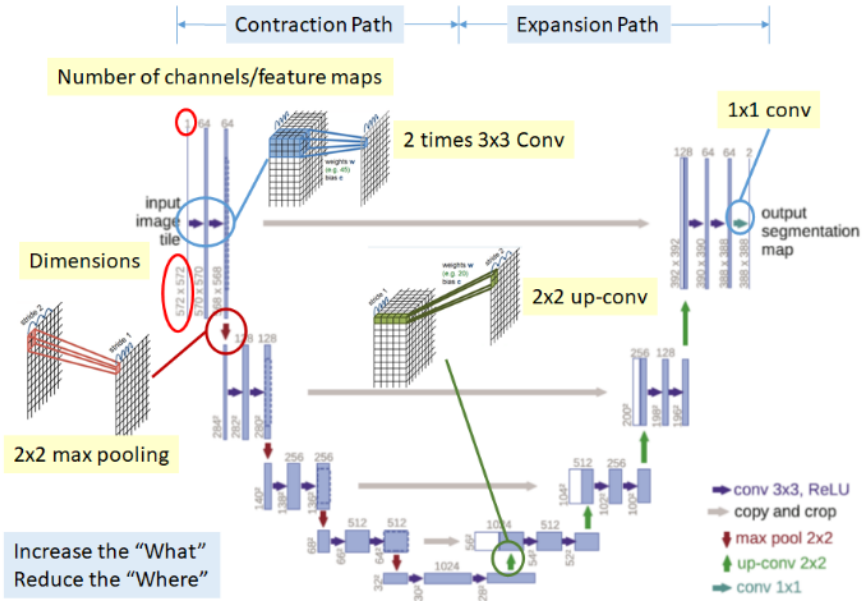


Figure 3.13: Illustrative 3 Dimensional U-Net

By incorporating Bayesian principles and variational inference techniques into the U-Net architecture, researchers have made significant progress in addressing the challenge of uncertainty estimation. Bayesian 3D U-Nets provide a more robust and reliable framework for segmentation tasks, especially in scenarios where accurate uncertainty estimates are essential for decision-making and confidence assessment.

3.8 Loss Function

In deep learning, the choice of an appropriate loss function is crucial for training a model effectively. When it comes to segmentation tasks, where the goal is to assign a label to each pixel or voxel in an input image or volume, specialized loss functions are commonly used. These loss functions are designed to measure the dissimilarity between the predicted segmentation and the ground truth, guiding the model to learn accurate segmentations.

One commonly used loss function for segmentation tasks is the *pixel-wise cross-entropy loss*, also known as *binary cross-entropy loss* or *softmax cross-entropy loss*. This loss function is well-suited for binary segmentation, where each pixel is classified as either foreground or background. The pixel-wise cross-entropy loss calculates the average cross-entropy loss over all pixels in the image.

Let's denote the predicted segmentation map as $\mathbf{P}$ and the ground truth segmentation map as $\mathbf{G}$. Both $\mathbf{P}$ and $\mathbf{G}$ are matrices of the same size as the input image, with each element representing the predicted or ground truth label of a pixel. For

binary segmentation, the labels are typically encoded as 0 for background and 1 for foreground.

The pixel-wise cross-entropy loss is defined as follows:

$$\text{CrossEntropy}(\mathbf{P}, \mathbf{G}) = -\frac{1}{N} \sum_{i=1}^{N} [\mathbf{G}_i \log(\mathbf{P}_i) + (1 - \mathbf{G}_i) \log(1 - \mathbf{P}_i)]$$

where N is the total number of pixels, $\mathbf{G}_i$ is the ground truth label of the i-th pixel, and $\mathbf{P}_i$ is the predicted probability of the i-th pixel belonging to the foreground class.

This loss function penalizes the model for incorrect predictions, encouraging it to produce high probabilities for the correct class and low probabilities for the incorrect class. The logarithmic term ensures that the loss is higher when the predicted probability deviates further from the ground truth label. By minimizing this loss, the model learns to generate segmentation maps that closely match the ground truth.

In addition to the pixel-wise cross-entropy loss, there are other loss functions commonly used in segmentation tasks, depending on the specific requirements and characteristics of the problem. For instance, when dealing with multi-class segmentation, where each pixel can be assigned one of multiple class labels, the *categorical cross-entropy loss* is often employed. This loss extends the binary cross-entropy loss to handle multiple classes.

Furthermore, for tasks where spatial coherence and smoothness are desired in the predicted segmentation, additional loss terms such as the *dice loss* or the *Jaccard loss* can be incorporated. These losses measure the overlap between the predicted and ground truth segmentations, encouraging the model to produce more accurate and coherent segmentations.

3.8.1 Dice Loss

The Dice loss is another commonly used loss function in segmentation tasks. It measures the overlap or similarity between the predicted and ground truth segmentations. The Dice loss is particularly useful when dealing with imbalanced datasets, where the number of background pixels far exceeds the number of foreground pixels.

The Dice loss is defined as follows:

$$\text{DiceLoss}(\mathbf{P}, \mathbf{G}) = 1 - \frac{2 \sum_{i=1}^{N} \mathbf{P}_i \mathbf{G} i + \epsilon}{\sum i = 1^N \mathbf{P} i^2 + \sum i = 1^N \mathbf{G}_i^2 + \epsilon}$$

where ϵ is a small constant added for numerical stability. The Dice loss ranges from 0 to 1, with 0 indicating no overlap between the predicted and ground truth segmentations, and 1 indicating perfect overlap.

The Dice loss encourages the model to generate segmentations that have high overlap with the ground truth, promoting accurate segmentation of both foreground and background regions. It is particularly effective when the foreground class is small compared to the background class.

In practice, a common approach is to combine the cross-entropy loss and the Dice loss to create a hybrid loss function. This combination takes advantage of the

benefits of both loss functions, encouraging accurate pixel-wise classification while also promoting spatial coherence and overlap with the ground truth.

Choosing an appropriate loss function depends on the specific requirements and characteristics of the segmentation task. It is common to experiment with different loss functions

3.8.2 Cross Entropy Loss

One commonly used loss function for segmentation tasks is the *cross-entropy loss*, also known as the *softmax cross-entropy loss*. This loss function is well-suited for multi-class segmentation, where each pixel can be assigned one of the multiple class labels.

Let's denote the predicted segmentation map as $\mathbf{P}$ and the ground truth segmentation map as $\mathbf{G}$. Both $\mathbf{P}$ and $\mathbf{G}$ are matrices of the same size as the input image, with each element representing the predicted or ground truth label of a pixel. The cross-entropy loss is defined as follows:

$$\text{CrossEntropy}(\mathbf{P}, \mathbf{G}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} \mathbf{G}ic \log(\mathbf{P}ic)$$

where N is the total number of pixels, C is the number of classes, $\mathbf{G}ic$ is the ground truth label of the i-th pixel for the c-th class, and $\mathbf{P}ic$ is the predicted probability of the i-th pixel belonging to the c-th class. This loss function penalizes the model for incorrect predictions and encourages it to produce high probabilities for the correct class.

3.8.3 BEC-DICE Loss

In the field of segmentation, it is crucial to employ an effective loss function that captures important features and preserves information in the spatial domain. In this regard, we utilize a combination of the Binary Cross-Entropy (BCE) loss and the Dice loss, referred to as the BCE-Dice loss. This hybrid loss function allows us to leverage both spatial and label-wise information for accurate segmentation.

1. **Binary Cross-Entropy (BCE) Loss:** The BCE loss measures the disparity between the predicted probability and the ground truth label. It is commonly employed in binary classification tasks. The BCE loss is calculated by taking the negative logarithm of the predicted probability for the correct label. The formulation is given by Equation 3.17.

2. **Dice Loss:** The Dice loss quantifies the overlap between the predicted segmentation and the ground truth segmentation. It is computed by evaluating the ratio of twice the intersection of the two segmentations to the sum of the pixels in both segmentations. The Dice loss aids in evaluating the similarity between the predicted and ground truth segmentations. The mathematical expression is provided by Equation 3.18.

By combining the BCE loss and the Dice loss, the BCE-Dice loss enables us to capture both spatial and label-wise information during the segmentation task. The

BCE loss encourages the model to correctly classify the pixels, while the Dice loss motivates the model to accurately segment the regions.

The BCE-Dice loss is formulated as the sum of the BCE loss and the Dice loss, as shown in Equation (5):

$$\text{BCE-Dice Loss} = \text{BCE} + \text{Dice Loss} \tag{3.16}$$

where:

$$\text{BCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{3.17}$$

$$\text{Dice Loss}(y, \hat{y}) = 1 - \frac{2\sum_{i=1}^{N} y_i \hat{y}_i + \epsilon}{\sum_{i=1}^{N} y_i + \sum_{i=1}^{N} \hat{y}_i + \epsilon} \tag{3.18}$$

In the above equations, y represents the ground truth, $\hat{y}$ denotes the predicted output, and ϵ is a small value added to prevent division by zero errors.

The BCE-Dice loss provides a comprehensive and effective approach for segmentation tasks, allowing the model to capture both local and global information, resulting in more accurate and reliable segmentation results.

3.9 Transformers

Transformers [47] has emerged as a powerful architecture in the field of AI, particularly in large language models (LLMs). They have made significant advancements in various applications such as text generation and image synthesis. Companies like Meta FAIR, OpenAI, and DeepMind have dedicated research departments to explore and enhance the capabilities of LLMs, with the goal of integrating them into robotics and multi-modal interaction.

At a high level, Transformers are encoder-decoder architectures that excel in mapping data representations across domains and reconstructing data. Initially developed for translation tasks, the Transformers architecture consists of an encoder and a decoder. The encoder processes the input data and produces a fixed-length vector representation, while the decoder takes this vector and generates the output sequence. The models are trained together to maximize the conditional log-likelihood of the output given the input.

In the original Transformer architecture, both the encoder and decoder consisted of six identical layers. Each layer in the encoder had two sub-layers: a multi-head self-attention layer and a feed-forward network. The self-attention layer computed the output representation for each input token based on all input tokens. Residual connections and layer normalization were applied to each sub-layer. The output representation size of the encoder was 512.

In the decoder, the multi-head self-attention layer was modified to include a masking mechanism. This masking ensured that the decoder could only attend to tokens preceding the one it was trying to predict. Additionally, a third sub-layer was introduced in the decoder, which involved multi-head attention over the encoder's outputs. It is important to note that these specific details have been modified in various Transformer variations, such as BERT and GPT, which may focus solely on the encoder or decoder components 3.14

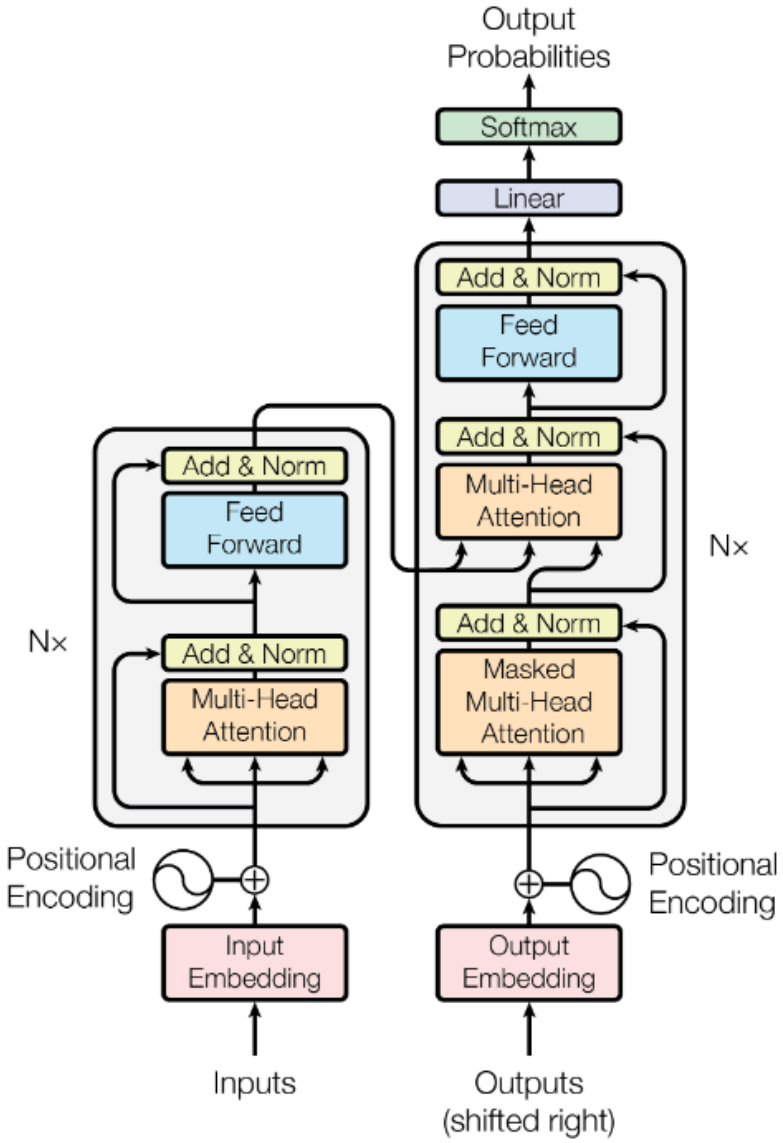


Figure 3.14: Transfomers Model Components

Transformers have revolutionized the field of AI by providing a flexible and powerful architecture for handling sequential data and language tasks. Their ability to capture dependencies between different elements of a sequence has led to significant improvements in natural language processing and other related domains.

3.9.1 Self-Attention

Self-Attention is a mechanism used in neural networks, particularly in the Transformer model, to capture the importance of different words in a sentence or sequence. The mathematical formula for Self-Attention is as follows:

$$\text{Self-Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

We think of the attention mechanism as a block in a neural network that allows

us to focus more on relevant features in the text by assigning higher probabilities to attend to previous or subsequent words. the Self-Attention Mechanism (SAM) was created to improve sequential modeling performance by addressing the limitations of recurrent and long-term memory in models such as LSTM and RNN. One of the key achievements of SAM is the parallel processing of sequences, achieved by dividing the process into Query, Key, and Value components. To illustrate how SAM works, let's consider the analogy of a database search using a look-up table. you might wonder why the Query, Key, and Value have different meanings but still receive the same input data. This is where SAM comes into play. It finds similarities between the Query (Q) and Key (K), and then uses this information to operate on the Value (V). Now, let's define the SAM formula in more depth.

Attention Mechanism "Look at Each Other" in the Encoder,given a sequence of input tokens $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where each $\mathbf{x}_i \in \mathbb{R}^d$, for $1 \leq i \leq n$, the self-attention mechanism outputs a sequence of the same length, $\mathbf{y}_1, \ldots, \mathbf{y}_n$. The learning process is represented by the following equation:

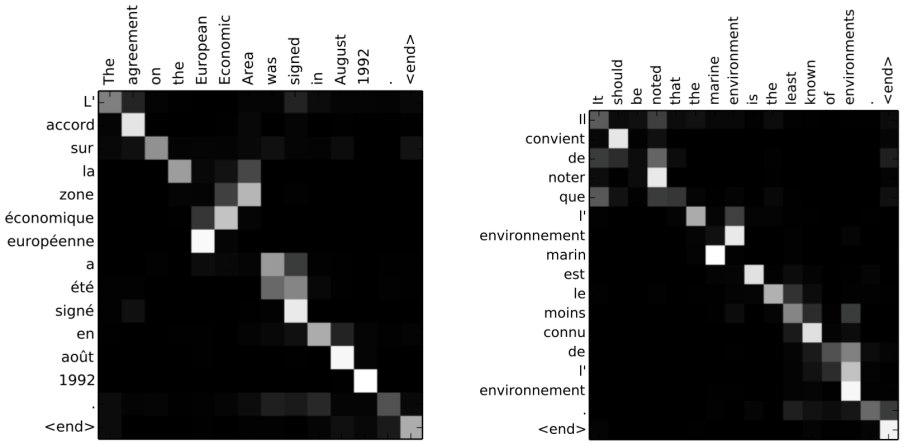


Figure 3.15: Sequence of input tokens

Theorem 3.9.1 (Self-Attention)**.** *The Self-Attention mechanism is defined as follows:*

$$\textit{Self-Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \textit{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

Lemma 3.9.2. *Consider a sequence of input tokens $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where each $\mathbf{x}_i \in \mathbb{R}^d$, for $1 \leq i \leq n$. The Self-Attention mechanism outputs a sequence of the same length, $\mathbf{y}_1, \ldots, \mathbf{y}_n$, and can be summarized as follows:*

1. **Initialize Learnable Weights:** We create three linear projections of each embedding $\mathbf{e}_i$ using three learned matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$:

$$\mathbf{Q}_i = \mathbf{W}_Q \mathbf{e}_i$$
$$\mathbf{K}_i = \mathbf{W}_K \mathbf{e}_i$$
$$\mathbf{V}_i = \mathbf{W}_V \mathbf{e}_i$$

   These projections are referred to as the Query, Key, and Value matrices, respectively.

2. **Dot-Product Similarity:** We compute the dot-product similarity between the Query ($\mathbf{Q}$) and Key ($\mathbf{K}$) matrices:

$$\text{Similarity}(\mathbf{Q}, \mathbf{K}) = \mathbf{Q}\mathbf{K}^T$$

This results in a square matrix of size $n \times n$.

3. **Attention Scoring Functions:**

$$\mathbf{S} = \text{softmax}(\text{Similarity}(\mathbf{Q}, \mathbf{K})) = \text{softmax}(\mathbf{Q}\mathbf{K}^T)$$

The softmax function ensures that the weights for each Value are positive and sum to 1.

4. **Weighted Sum:** We compute the weighted sum of the Value matrix using the softmax weights:

$$\text{Weighted Sum}(\mathbf{V}, \mathbf{S}) = \mathbf{S}\mathbf{V}$$

This results in a matrix of size $n \times d$, where d is the number of features in each Value vector.

5. **Output:** We obtain the final output matrix by concatenating the Weighted Sum matrices computed for each embedding in the input sequence:

$$\text{Output} = [\text{Weighted Sum}(\mathbf{V}_1, \mathbf{S}), \text{Weighted Sum}(\mathbf{V}_2, \mathbf{S}), \ldots, \text{Weighted Sum}(\mathbf{V}_n, \mathbf{S})]$$

In practice, the input embeddings are typically combined with positional encoding to capture the sequential information of the input sequence.

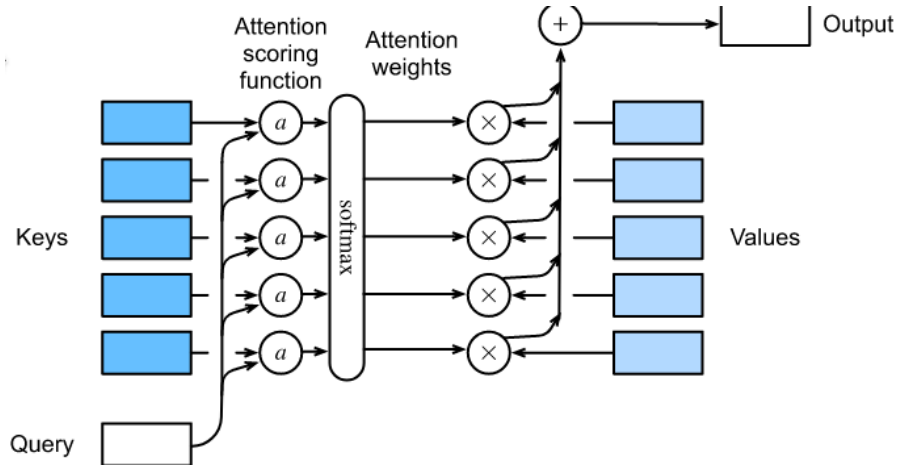


Figure 3.16: Sequence of input tokens

The Self-Attention mechanism, with its Query, Key, and Value components and learnable weight matrices, allows the neural network to attend to relevant features and capture important dependencies within the input sequence.

3.9.2 Multi-Head Self-Attention

In the Transformer, the Attention module repeats its computations multiple times in parallel. Each of these is called an Attention Head. The Attention module splits its Query, Key, and Value parameters N-ways and passes each split independently through a separate Head. All of these similar Attention calculations are then combined together to produce a final Attention score. This is called Multi-Head Attention and gives the Transformer greater power to encode multiple relationships and nuances for each word.

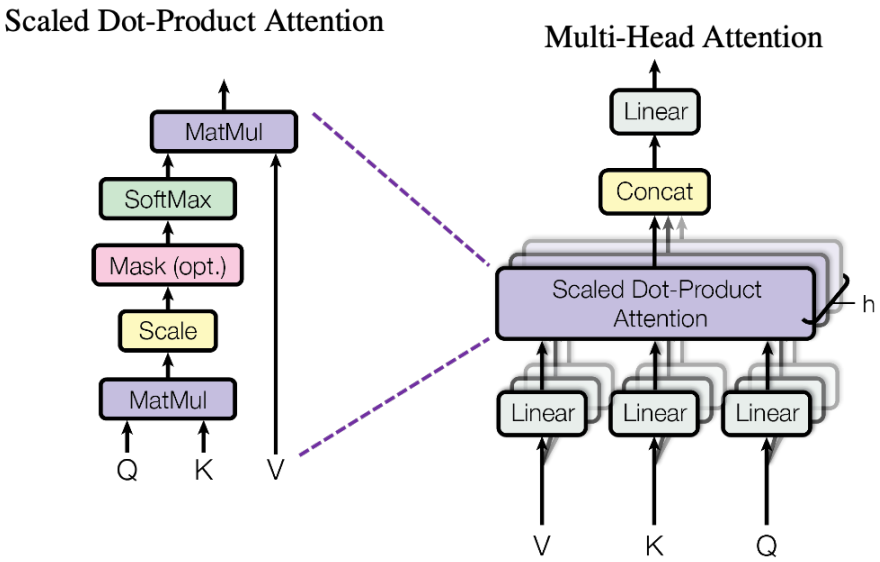


Figure 3.17: Multi-Head Self-Attention

The motivation behind Multi-Head Attention is to make the model focus on different aspects of the input sequence. Instead of using a single attention mechanism, Multi-Head Attention has multiple "heads" that work independently.

Multi-Head Attention allows the Transformer model to attend to different aspects of the input sequence simultaneously, capturing multiple relationships and nuances. It enhances the model's ability to process complex patterns and improves its performance in various natural language processing tasks.

Lemma 3.9.3. *Multi-Head Attention allows the Transformer model to attend to different aspects of the input sequence simultaneously, capturing multiple relationships and nuances. It enhances the model's ability to process complex patterns and improves its performance in various natural language processing tasks.*

Proof. Let $\mathbf{E} = [\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_n}]$ be a sequence of n embeddings.

We create h sets of learnable projection matrices $\mathbf{W_Q^h}$, $\mathbf{W_K^h}$, and $\mathbf{W_V^h}$, where h is the number of attention heads.

For each head h, we compute the Query ($\mathbf{Q^h}$), Key ($\mathbf{K^h}$), and Value ($\mathbf{V^h}$) matrices as follows:

1. Compute the Query matrix:

$$\mathbf{Q_i^h} = \mathbf{W_Q^h e_i}$$

2. Compute the Key matrix:

$$\mathbf{K_i^h} = \mathbf{W_K^h e_i}$$

3. Compute the Value matrix:

$$\mathbf{V_i^h} = \mathbf{W_V^h e_i}$$

We then compute the dot-product similarity between the Query and Key matrices:

$$\text{Similarity}(\mathbf{Q^h}, \mathbf{K^h}) = \frac{\mathbf{Q^h K^{h^T}}}{\sqrt{d_k}}$$

where d_k is the dimension of the Key matrix.
We apply a softmax function to the Similarity matrix:

$$\mathbf{S^h} = \text{softmax}(\text{Similarity}(\mathbf{Q^h}, \mathbf{K^h})) = \text{softmax}\left(\frac{\mathbf{Q^h K^{h^T}}}{\sqrt{d_k}}\right)$$

We compute the weighted sum of the Value matrix using the softmax weights:

$$\text{Weighted Sum}(\mathbf{V^h}, \mathbf{S^h}) = \mathbf{S^h V^h}$$

We concatenate the output matrices from all heads together:

$$\text{Output} = [\text{Weighted Sum}(\mathbf{V^1}, \mathbf{S^1}), \text{Weighted Sum}(\mathbf{V^2}, \mathbf{S^2}), \ldots, \text{Weighted Sum}(\mathbf{V^h}, \mathbf{S^h})]$$

This results in a matrix of size $n \times d_v h$, where d_v is the number of features in each Value vector.

The multi-head self-attention mechanism allows the model to capture different types of relationships between words or tokens in the input sequence and effectively encode them, leading to improved performance in various natural language processing tasks. □

3.10 Attention Networks in Segmentation Task

Attention networks were initially introduced in the field of Natural Language Processing (NLP). Researchers discovered that dealing with long sentences posed a challenge for models. Therefore, it is more effective if the model can focus its attention on specific words rather than the entire sentence. Attention was first proposed for neural machine translation in works by [48] and [49]. For instance, when translating the English sentence "I am a student" to French as "Je suis étudiant," it can be observed that while the model should output "Je," it only needs to attend to the word "I" rather than the entire sentence. Similarly, to output "étudiant," it should attend to the words "a" and "student." Attention networks have also been applied to image captioning problems, where the network takes an image as input and generates a descriptive sentence as output. The work "Show, Attend and Tell" [50]introduced attention networks for image captioning, which learned to attend to specific regions while generating corresponding words Figure.3.18. In general, attention networks emphasize informative features and suppress less informative ones.

Types of Attention There are primarily two types of attention mechanisms: soft attention and hard attention. In soft attention, the network distributes its attention to all regions with certain weights. On the other hand, hard attention selects only one region while disregarding others. Soft attention networks can be trained end-to-end and are differentiable, whereas hard attention networks cannot be trained end-to-end and rely on reinforcement learning techniques. Therefore, this report will primarily focus on soft attention networks.

Interpretable and Informative One attractive feature of attention networks is that they provide insights into what is happening inside the network and the circumstances that led the network to predict a certain class. Attention networks reveal which regions or features the model is attending to, providing interpretability and transparency.

Figure 3.18: xample of how attention networks work in image captioning, reproduced with permission

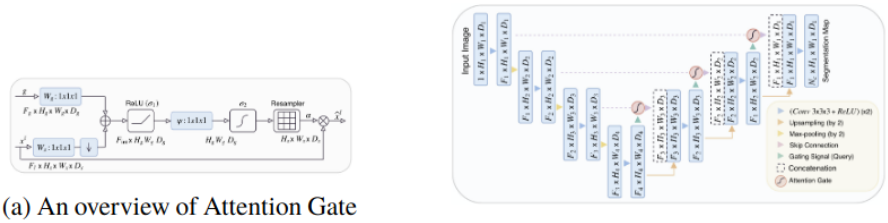
In this section, we have introduced attention networks and discussed their relevance in the segmentation task. We have provided examples of attention networks in neural machine translation and image captioning problems. The subsequent sections will explore different categories of attention networks that have been used for semantic segmentation. These categories are based on how attention is generated and incorporated into the main network. It is important to note that these categories are proposed by the author as a conceptual map to understand attention networks, and following our study we focus on some Attention we based on to build our method in the next section we will dive into some of the used attention mechanism widely used in the Segmentation task

3.10.1 Self-Designed Attention Added to the Decoder of a Model

Attention gate (AG) [19] is an attention model that focuses on target structures of varying shapes and sizes for medical image analysis. The authors justify the use of attention gates integrated with CNN models (e.g., U-Net [45]) by highlighting their ability to learn to suppress irrelevant regions and emphasize salient ones. This eliminates the need for an external localization module, resulting in computational savings and increased model efficiency. The attention gate utilizes high-level contextual information to weight low-level features based on their location. This weighted information is then passed through a non-linearity and normalized. Figure.3.19b provides an overview of the AG model.

To integrate the attention gate with U-Net [45], the features from the decoder path are used as high-level features to weigh the low-level features from the encoder. The resulting weighted features are then added to the next decoding layer. Fig-

ure.3.19b illustrates the integration of the attention gate with U-Net . The authors conducted experiments using two large 3D CT abdominal datasets and evaluated the performance using metrics such as the Dice coefficient, precision, recall, and surface distance.



(a) An overview of Attention Gate model

(b) Integration of AG with U-Net

Figure 3.19: Attention Gate model and its integration with U-Net, reproduced with permission

By incorporating the self-designed attention gate into the U-Net architecture, the model can effectively attend to relevant regions and suppress irrelevant ones, leading to improved segmentation performance in medical image analysis tasks.

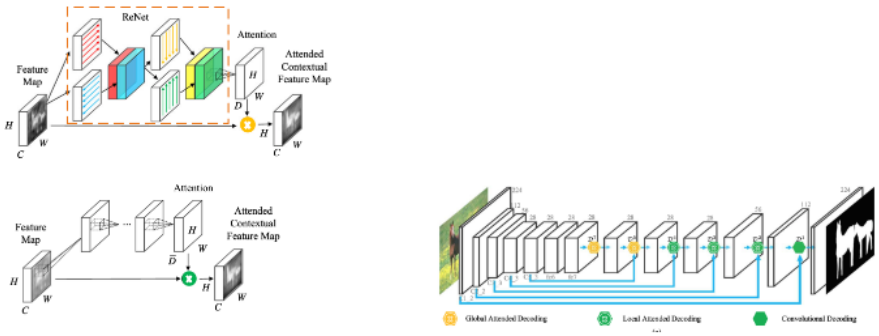
Pixel-wise Contextual Attention Network (PiCANet) [51] is designed to selectively attend to informative context locations for each pixel. PiCANet is formulated in both global and local forms to attend to global and local contexts. The PiCANet module is integrated with U-Net [44]. In the U-Net architecture, a feature map from the encoder is concatenated with a feature map from the decoder and passed to either the global or local PiCANet module. The output is then passed to the next decoder layer. Figure 3.20b illustrates the integration of PiCANet with U-Net.

For the global PiCANet, the aim is to allow each pixel to perceive the overall feature map. To achieve this, four recurrent neural networks (RNNs) are employed to sweep the image horizontally and vertically in both directions, incorporating the global context. The hidden states of each pixel are concatenated, enabling each pixel to capture surrounding contexts. Finally, a convolutional layer with softmax normalization is used to shape the output to the desired form.

For the local PiCANet, the goal is to enable each pixel to perceive a local context region of size $\bar{W} \times \bar{W}$. Convolutional layers are used to achieve this purpose. Similar to the global PiCANet, a convolutional layer with softmax normalization is employed to shape the output.

Figure A.13a provides a graphical representation of the global and local PiCANet modules. The authors evaluated their approach using six different datasets, including PASCAL VOC 2010 [52], and used metrics such as precision, recall, weighted F-score, and Mean Absolute Error (MAE) for evaluation.

By incorporating the PiCANet modules into the U-Net architecture, the model can effectively attend to both global and local contexts, allowing for more precise and informative pixel-wise predictions in the semantic segmentation task.



(a) Global and Local PiCANet

(b) Integration of PiCANet with U-Net

Figure 3.20: PiCANet and its integration , reproduced with permission

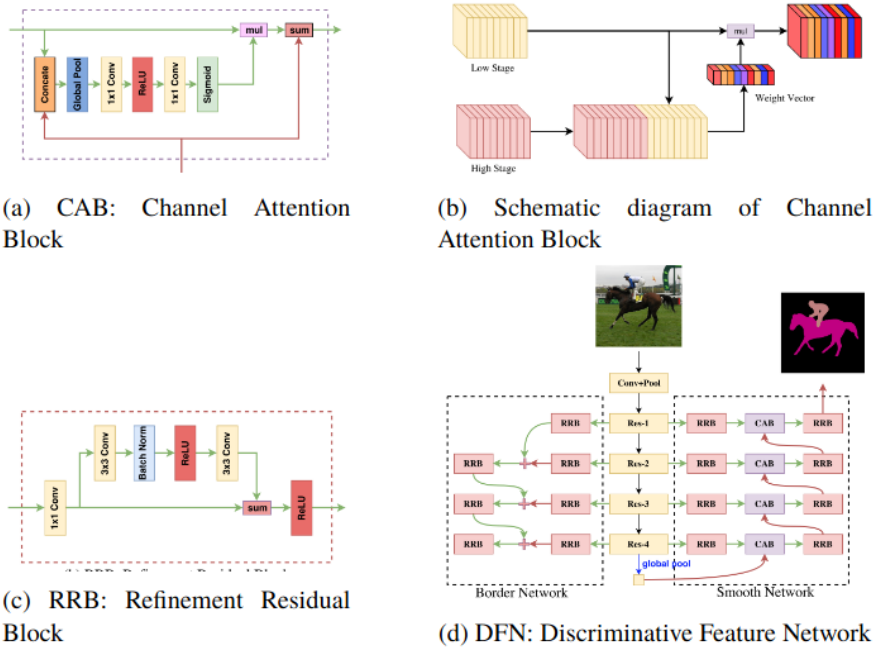
In [51], the authors addressed two challenges in semantic segmentation: intra-class inconsistency and inter-class indistinction. To tackle these challenges, they proposed a Discriminative Feature Network (DFN) [1], which consists of two sub-networks: the Smooth Network and the Border Network.

The Smooth Network aims to address intra-class inconsistency by capturing a multi-scale context. It utilizes a U-shaped architecture and incorporates Channel Attention Blocks (CAB) to guide low-level features with spatial predictions based on high-level features and semantic predictions. The CAB employs global average pooling to capture global context information. Figures 3.21a and 3.21b provide graphical explanations of the Channel Attention Block.

The Border Network focuses on inter-class indistinction and leverages low-stage information for accurate edge detection and high-stage information for capturing semantic information. It emphasizes the semantic boundary that separates classes. The network incorporates a Refinement Residual Block (RRB) for this purpose. Figure 3.21c shows a graphical explanation of the RRB used in the DFN architecture.

The authors conducted experiments using the PASCAL VOC 2012 [52] and Cityscapes [53] datasets and evaluated the performance using the mean Intersection over Union (mIoU) metric.

By incorporating the Smooth Network and the Border Network into the DFN architecture, the model addresses both intra-class inconsistency and inter-class indistinction, leading to improved semantic segmentation performance.



(a) CAB: Channel Attention Block

(b) Schematic diagram of Channel Attention Block

(c) RRB: Refinement Residual Block

(d) DFN: Discriminative Feature Network

Figure 3.21: DFN and its components [1], reproduced with permission.

3.11 Fourier Transform

The Fourier Transform is a mathematical technique that decomposes a function into its frequency components. It allows us to analyze and represent signals and functions in the frequency domain, which provides valuable insights into their underlying properties. This section focuses on the 2D Fourier Transform.

The Fourier Theorem states that any periodic function can be expressed as a sum of sine and cosine waves with appropriate amplitudes and phases. This theorem is named after Joseph Fourier, a French mathematician who made significant contributions to the study of heat transfer and vibrations. The Fourier series, derived from this theorem, has many applications in various fields of mathematics.

In the context of the Fourier Transform, a continuous function $f(x)$ that is integrable over $\mathbb{R}$ can be represented as an infinite sum of sines and cosines. This representation is known as the Fourier series and is given by:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi nx}{T}\right) + b_n \sin\left(\frac{2\pi nx}{T}\right) \right]$$

Here, T represents the period of the function, and the Fourier coefficients a_0, a_n, and b_n can be calculated using the following formulas:

$$a_0 = \frac{1}{T}\int_{-T/2}^{T/2} f(x), dx \; a_n \; = \frac{2}{T}\int_{-T/2}^{T/2} f(x)\cos\left(\frac{2\pi nx}{T}\right), dx \; b_n = \frac{2}{T}\int_{-T/2}^{T/2} f(x)\sin\left(\frac{2\pi nx}{T}\right), dx$$

On the other hand, the inverse Fourier transform allows us to reconstruct the original function $f(x)$ from its Fourier coefficients. It is given by:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i2\pi nx/T}$$

Here, c_n represents the Fourier coefficients and can be calculated using the formula:

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(x) e^{-i2\pi nx/T}, dx$$

The Fourier transform and its inverse are fundamental tools in signal processing, image processing, and various other areas of science and engineering. They allow us to analyze and manipulate signals and images in the frequency domain, providing insights into their frequency components and facilitating tasks such as noise removal, compression, and feature extraction.

Note that the presented equations and concepts refer specifically to the 2D Fourier Transform, which is applicable to two-dimensional signals and functions, such as images.

3.11.1 Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is the equivalent of the continuous Fourier Transform for signals known only at discrete time instants separated by sample times. In contrast to the continuous Fourier Transform, which operates on continuous signals defined over an infinite interval, the DFT processes finite sequences of data.

Consider a continuous signal, denoted as $x(t)$, which serves as the source of the data. Let the samples of this signal be represented as $x[n]$, where n indicates the sample index. The Fourier Transform of the original signal, $x(t)$, would be expressed as:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t}, dt$$

Now, with discrete data points, we can consider each sample $x[n]$ as an impulse with an area $x[n]\Delta t$. As a result, we can rewrite the Fourier Transform equation for discrete signals as:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-j\omega n \Delta t}$$

The DFT enables us to compute the discrete frequency spectrum $X(k)$ of the signal, where k represents the frequency index ranging from 0 to $N-1$. The DFT formula calculates the contribution of each sample at different discrete frequencies.

To efficiently compute the DFT, the Fast Fourier Transform (FFT) algorithm is widely used. The FFT reduces the computational complexity of the DFT from $O(N^2)$ to $O(N \log N)$, making it practical for real-time applications and large data sets.

By performing the DFT, we gain insights into the frequency content of the signal. The frequency spectrum reveals the amplitudes and phases of different frequency

components present in the signal. This information is utilized in various fields, including digital signal processing, image processing, audio analysis, telecommunications, and scientific research.

It is important to note that the DFT assumes periodicity of the signal with a period of N samples. When applying the DFT to non-periodic signals, spectral leakage and other artifacts may arise. Techniques like windowing can be employed to minimize these effects.

The inverse DFT allows us to reconstruct the original signal from its frequency spectrum. It is given by:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j\omega n \Delta t}$$

The DFT and its inverse play a fundamental role in digital signal processing. They provide a powerful tool for analyzing, manipulating, and synthesizing signals in the frequency domain.

3.11.2 Fast Fourier Transform

The Fast Fourier Transform (FFT) is a widely used algorithm for efficiently computing the Discrete Fourier Transform (DFT) of a sequence. In the context of medical imaging, the FFT plays a crucial role in various applications, such as image filtering, registration, and analysis. By leveraging the FFT, we can efficiently analyze the frequency components of medical images, enabling advanced processing and interpretation.

The 2D Fast Fourier Transform (FFT) extends the concept of the 1D FFT to two-dimensional signals, such as images. Let $f(x, y)$ be a 2D image with dimensions $M \times N$. The 2D FFT of $f(x, y)$, denoted as $F(u, v)$, is computed using the following equation:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

Here, $F(u, v)$ represents the Fourier Transform of the 2D image $f(x, y)$, and (u, v) are the frequency domain variables. The 2D FFT allows us to analyze the spatial frequency content of the image.

The inverse 2D Fast Fourier Transform (IFFT) enables us to reconstruct the original image $f(x, y)$ from its frequency domain representation $F(u, v)$. It is defined as follows:

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

The 2D FFT and IFFT are powerful tools in medical imaging. They allow us to analyze the spatial frequency components of an image, identify patterns, filter out noise, and extract relevant information. By leveraging the frequency domain, we can apply various image processing techniques, such as image enhancement, restoration, and compression, to improve the quality and interpretability of medical images.

In this section, we will explore the applications of Fast Fourier Transform (FFT) in medical imaging. We begin by providing an explanation of Fourier Transform and Discrete Fourier Transform (DFT), which will lay the foundation for understanding FFT. We will then discuss the advantages of utilizing FFT over convolution operations, highlighting the fact that both methods yield equivalent results in different domains.

The 2D Fast Fourier Transformation (FFT) involves processing a 2D image, denoted as $f(x, y)$, with dimensions $M \times N$. The FFT of $f(x, y)$ can be defined as follows:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi(ux/M+vy/N)} \tag{3.19}$$

Here, $F(u, v)$ represents the Fourier Transform of $f(x, y)$.

Conversely, the inverse 2D Fast Fourier Transform (IFFT) of $F(u, v)$ is given by:

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{i2\pi(ux/M+vy/N)} \tag{3.20}$$

3.11.3 Fast Fourier Transform for Medical Imaging

In this equation, $f(x, y)$ corresponds to the original 2D image. It is worth noting that the 2D FFT and IFFT are extensions of their 1D counterparts, enabling computations in two dimensions. These transformations find extensive applications in image processing and computer vision.

The Convolution Theorem establishes a fundamental connection between convolution and the Fourier Transform. Let's consider two continuous functions, $f(x)$ and $g(x)$, with Fourier Transforms $F(\omega)$ and $G(\omega)$, respectively. The convolution of $f(x)$ and $g(x)$ can be defined as follows:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x - t) g(t) dt \tag{3.21}$$

The Fourier Transform of the convolution is given by:

$$\mathcal{F} f(x) * g(x) = F(\omega) G(\omega) \tag{3.22}$$

In other words, convolution in the time domain corresponds to multiplication in the frequency domain.

To establish the equivalence of convolution and the Fourier Transform, let's consider the convolution of two functions, f and g, in the time domain:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{3.23}$$

The Fourier Transform of a function $f(t)$ is given by:

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{3.24}$$

By employing the Fourier transform, we can demonstrate the equivalence of convolution and multiplication in the frequency domain. Starting with the Fourier transform of the convolution of f and g:

$$\mathcal{F}f * g(\omega) = \int_{-\infty}^{\infty} (f * g)(t) e^{-i\omega t} dt \tag{3.25}$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau e^{-i\omega t} dt \tag{3.26}$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau) g(t-\tau) e^{-i\omega t} dt d\tau \tag{3.27}$$

$$= \int_{-\infty}^{\infty} f(\tau) \int_{-\infty}^{\infty} g(t-\tau) e^{-i\omega t} dt d\tau \tag{3.28}$$

$$= \int_{-\infty}^{\infty} f(\tau) \left(\int_{-\infty}^{\infty} g(t-\tau) e^{-i\omega t} dt \right) d\tau \tag{3.29}$$

$$= \int_{-\infty}^{\infty} f(\tau) e^{-i\omega\tau} \left(\int_{-\infty}^{\infty} g(u) e^{-i\omega u} du \right) d\tau \tag{3.30}$$

$$= \hat{f}(\omega)\hat{g}(\omega) \tag{3.31}$$

Consequently, convolution in the time domain is equivalent to multiplication in the frequency domain.

In summary, the Fast Fourier Transform (FFT) is a valuable tool in medical image processing. By utilizing the concepts of Fourier Transform and convolution, the FFT enables efficient computation and provides results equivalent to traditional convolution operations in a different domain.

When considering the computational cost of image processing algorithms, it is important to analyze their algorithmic complexity. In this section, we compare the algorithm complexity of Fast Fourier Transform (FFT) and convolution operations.

Operation	**Algorithm Complexity**
FFT	$\mathcal{O}(N \log N)$
Convolution	$\mathcal{O}(N^2)$

Table 3.1: Algorithm Complexity of FFT and Convolution Operations

The table above presents the algorithm complexity of FFT and convolution operations. The algorithm complexity is often represented using Big O notation, which provides an upper bound on the computational resources required.

For FFT, the algorithm complexity is $\mathcal{O}(N \log N)$, where N represents the number of elements in the input signal. The FFT algorithm exploits the divide-and-conquer approach, utilizing a combination of decimation and butterfly operations to reduce the computational complexity significantly compared to direct computation of the Discrete Fourier Transform (DFT), which has a complexity of $\mathcal{O}(N^2)$.

On the other hand, convolution operations have an algorithm complexity of $\mathcal{O}(N^2)$. Convolution involves a pairwise multiplication and summation of corresponding elements from two signals, requiring N^2 operations to compute the output. This makes convolution computationally expensive, especially for larger input sizes.

The significant difference in algorithm complexity between FFT and convolution operations highlights the advantage of FFT in terms of computational efficiency. By leveraging FFT, medical image processing algorithms can achieve substantial speed-ups and reduce computational costs compared to traditional convolution operations.

It is important to note that the algorithm complexity presented here represents the theoretical analysis of these operations and may not account for additional factors such as memory access patterns or hardware optimizations. Nonetheless, the provided complexity values serve as a general guideline for understanding the comparative efficiency of FFT and convolution operations.

In conclusion, the FFT algorithm offers a more efficient computational approach for medical image processing compared to convolution operations, as evidenced by its lower algorithm complexity.

Bibliography

[1] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Learning a discriminative feature network for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1857–1866, 2018.

[2] Kevin Jamart, Zhaohan Xiong, Gonzalo D Maso Talou, Martin K Stiles, and Jichao Zhao. Mini review: Deep learning for atrial segmentation from late gadolinium-enhanced mris. *Frontiers in Cardiovascular Medicine*, 7:86, 2020.

[3] Alicia M Maceira, Juan Cosín-Sales, Michael Roughton, Sanjay K Prasad, and Dudley J Pennell. Reference left atrial dimensions and volumes by steady state free precession cardiovascular magnetic resonance. *Journal of cardiovascular magnetic resonance*, 12(1):1–10, 2010.

[4] Catalina Tobon-Gomez, Arjan J Geers, Jochen Peters, Jürgen Weese, Karen Pinto, Rashed Karim, Mohammed Ammar, Abdelaziz Daoudi, Jan Margeta, Zulma Sandoval, et al. Benchmark for algorithms segmenting the left atrium from 3d ct and mri datasets. *IEEE transactions on medical imaging*, 34(7):1460–1473, 2015.

[5] James F Sallis, Myron F Floyd, Daniel A Rodríguez, and Brian E Saelens. Role of built environments in physical activity, obesity, and cardiovascular disease. *Circulation*, 125(5):729–737, 2012.

[6] Xiangbin Liu, Liping Song, Shuai Liu, and Yudong Zhang. A review of deep-learning-based medical image segmentation methods. *Sustainability*, 13(3):1224, 2021.

[7] KKD Ramesh, G Kiran Kumar, K Swapna, Debabrata Datta, and S Suman Rajest. A review of medical image segmentation algorithms. *EAI Endorsed Transactions on Pervasive Health and Technology*, 7(27):e6–e6, 2021.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas 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*, pages 234–241. Springer, 2015.

[9] Getao Du, Xu Cao, Jimin Liang, Xueli Chen, and Yonghua Zhan. Medical image segmentation based on u-net: A review. *Journal of Imaging Science and Technology*, 2020.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas 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*, pages 234–241. Springer, 2015.

[11] Cheng Li, Baolong Guo, Geng Wang, Yan Zheng, Yang Liu, and Wangpeng He. Nice: Superpixel segmentation using non-iterative clustering with efficiency. *Applied Sciences*, 10(12):4415, 2020.

[12] Murong Wang, Xiabi Liu, Yixuan Gao, Xiao Ma, and Nouman Q Soomro. Superpixel segmentation: A benchmark. *Signal Processing: Image Communication*, 56:28–39, 2017.

[13] Zhaohan Xiong, Qing Xia, Zhiqiang Hu, Ning Huang, Cheng Bian, Yefeng Zheng, Sulaiman Vesal, Nishant Ravikumar, Andreas Maier, Xin Yang, et al. A global benchmark of algorithms for segmenting the left atrium from late gadolinium-enhanced cardiac magnetic resonance imaging. *Medical image analysis*, 67:101832, 2021.

[14] Catalina Tobon-Gomez, Arjan J Geers, Jochen Peters, Jürgen Weese, Karen Pinto, Rashed Karim, Mohammed Ammar, Abdelaziz Daoudi, Jan Margeta, Zulma Sandoval, et al. Benchmark for algorithms segmenting the left atrium from 3d ct and mri datasets. *IEEE transactions on medical imaging*, 34(7):1460–1473, 2015.

[15] Xiaoran Zhang, Michelle Noga, David Glynn Martin, and Kumaradevan Punithakumar. Fully automated left atrium segmentation from anatomical cine long-axis mri sequences using deep convolutional neural network with unscented kalman filter. *Medical image analysis*, 68:101916, 2021.

[16] Asma Kausar, Imran Razzak, Mohammad Ibrahim Shapiai, and Amin Beheshti. 3d shallow deep neural network for fast and precise segmentation of left atrium. *Multimedia Systems*, pages 1–11, 2021.

[17] Lei Li, Veronika A Zimmer, Julia A Schnabel, and Xiahai Zhuang. Medical image analysis on left atrial lge mri for atrial fibrillation studies: A review. *Medical image analysis*, 77:102360, 2022.

[18] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[19] Jo Schlemper, Ozan Oktay, Michiel Schaap, Mattias Heinrich, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention gated networks: Learning to leverage salient regions in medical images. *Medical image analysis*, 53:197–207, 2019.

[20] Chengjia Wang, Tom MacGillivray, Gillian Macnaught, Guang Yang, and David Newby. A two-stage u-net model for 3d multi-class segmentation on full-resolution cardiac data. In *Statistical Atlases and Computational Models of*

the Heart. Atrial Segmentation and LV Quantification Challenges: 9th International Workshop, STACOM 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers 9, pages 191–199. Springer, 2019.

[21] Ben Tsuda. *Towards understanding diseases of memory and learning: Artificial neural network models of memory formation, manipulation, and disease in the brain*. University of California, San Diego, 2022.

[22] Lequan Yu, Jie-Zhi Cheng, Qi Dou, Xin Yang, Hao Chen, Jing Qin, and Pheng-Ann Heng. Automatic 3d cardiovascular mr segmentation with densely-connected volumetric convnets. In *Medical Image Computing and Computer-Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part II 20*, pages 287–295. Springer, 2017.

[23] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016.

[24] Aliasghar Mortazi, Jeremy Burt, and Ulas Bagci. Multi-planar deep segmentation networks for cardiac substructures from mri and ct. In *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges: 8th International Workshop, STACOM 2017, Held in Conjunction with MICCAI 2017, Quebec City, Canada, September 10-14, 2017, Revised Selected Papers 8*, pages 199–206. Springer, 2018.

[25] Holger R Roth, Hirohisa Oda, Yuichiro Hayashi, Masahiro Oda, Natsuki Shimizu, Michitaka Fujiwara, Kazunari Misawa, and Kensaku Mori. Hierarchical 3d fully convolutional networks for multi-organ segmentation. *arXiv preprint arXiv:1704.06382*, 2017.

[26] Yongjie Duan, Jianjiang Feng, Jiwen Lu, and Jie Zhou. Context aware 3d fully convolutional networks for coronary artery segmentation. In *Statistical Atlases and Computational Models of the Heart. Atrial Segmentation and LV Quantification Challenges: 9th International Workshop, STACOM 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers 9*, pages 85–93. Springer, 2019.

[27] Shuman Jia, Antoine Despinasse, Zihao Wang, Hervé Delingette, Xavier Pennec, Pierre Jaïs, Hubert Cochet, and Maxime Sermesant. Automatically segmenting the left atrium from cardiac images using successive 3d u-nets and a contour loss. In *Statistical Atlases and Computational Models of the Heart. Atrial Segmentation and LV Quantification Challenges: 9th International Workshop, STACOM 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers 9*, pages 221–229. Springer, 2019.

[28] Fumin Guo, Matthew Ng, and Graham Wright. Cardiac mri left ventricle segmentation and quantification: A framework combining u-net and continuous

max-flow. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 450–458. Springer, 2018.

[29] Michael Markl, Alex Frydrychowicz, Sebastian Kozerke, Mike Hope, and Oliver Wieben. 4d flow mri. *Journal of Magnetic Resonance Imaging*, 36(5):1015–1036, 2012.

[30] Fangzhou Liao, Ming Liang, Zhe Li, Xiaolin Hu, and Sen Song. Evaluate the malignancy of pulmonary nodules using the 3-d deep leaky noisy-or network. *IEEE transactions on neural networks and learning systems*, 30(11):3484–3495, 2019.

[31] Wenjia Bai, Matthew Sinclair, Giacomo Tarroni, Ozan Oktay, Martin Rajchl, Ghislain Vaillant, Aaron M Lee, Nay Aung, Elena Lukaschuk, Mihir M Sanghvi, et al. Automated cardiovascular magnetic resonance image analysis with fully convolutional networks. *Journal of Cardiovascular Magnetic Resonance*, 20(1):1–12, 2018.

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[33] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[34] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[35] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

[36] Sam Wiseman, Sumit Chopra, Marc'Aurelio Ranzato, Arthur Szlam, Ruoyu Sun, Soumith Chintala, and Nicolas Vasilache. Training language models using target-propagation. *arXiv preprint arXiv:1702.04770*, 2017.

[37] Martin Riedmiller and A Lernen. Multi layer perceptron. *Machine Learning Lab Special Lecture, University of Freiburg*, pages 7–24, 2014.

[38] Maximilian Pfundstein. Human age prediction based on real and simulated rr intervals using temporal convolutional neural networks and gaussian processes, 2020.

[39] Simone Scardapane, Steven Van Vaerenbergh, Amir Hussain, and Aurelio Uncini. Complex-valued neural networks with nonparametric activation functions. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2):140–150, 2018.

[40] Ken Kreutz-Delgado. The complex gradient operator and the cr-calculus. *arXiv preprint arXiv:0906.4835*, 2009.

[41] Tohru Nitta. An extension of the back-propagation algorithm to complex numbers. *Neural Networks*, 10(8):1391–1415, 1997.

[42] Nicolò Benvenuto and Francesco Piazza. On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 40(4):967–969, 1992.

[43] George M Georgiou and Christos Koutsougeras. Complex domain backpropagation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):330–334, 1992.

[44] Akira Hirose. Continuous complex-valued back-propagation learning. *Electronics Letters*, 28(20):1854–1855, 1992.

[45] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016.

[46] Omkar Bhutra. Using deep learning to segmentcardiovascular 4d flow mri: 3d u-net for cardiovascular 4d flow mri segmentation and bayesian 3d u-net for uncertainty estimation, 2021.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[48] Felix Stahlberg. Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418, 2020.

[49] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[50] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[51] Nian Liu, Junwei Han, and Ming-Hsuan Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3089–3098, 2018.

[52] Sara Vicente, Joao Carreira, Lourdes Agapito, and Jorge Batista. Reconstructing pascal voc. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 41–48, 2014.

[53] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

A Appendix title

B Another Appendix