MIMESIS

UNIVERSITY OF STRASBOURG

MASTER CSMI

Internship Report : Innovative non-conformal finite element methods for augmented surgery

Authors:
Frédérique Lecourtier

Supervisors:
Michel Duprez
Emmanuel Franck

Inria

Date: June 30, 2023

Contents

PLAN :

1. Intro : Ce stage est un stage de M2, dans le cadre du master CSMI. C'est la suite d'un projet effèctué pendant le premier semestre.... explication rapide du projet (colab entre Cemosis et Mimesis dans le but de blablabla)
    (a) Présentation de Mimesis
    (b) Contexte : Mimesis est spécialisé dans la simulation en temps réel de blabla. C'est pourquoi ils ont développé une méthode appelée PhiFEM (explication des intérêts de PhiFEM).
    (c) Objectifs : On cherche ici à améliorer la précision de la solution ainsi que les temps pour l'obtenir en passant par le biais d'un FNO que l'on va entraîner avec des solutions PhiFEM qui sont bien adapté à ce type de réseau de neurones, due aux grilles cartésienne.
2. Finite Element Methods (FEM)
    (a) Standard FEM
    (b) PhiFEM
3. Fourier Neural Operator (FNO)
4. Correction :
   Présentation des différentes méthodes de Correction (2 cas tests : solution analytique, sol FNO) / Legendre / Modèle Dense ?
   On veut mettre un schéma qui explique Entrainement du FNO -> Sortie -> Correction (-> Legendre ?)
5. Résultats ?
6. Conclu
7. Bibliography
8. Appendix (Organisation of the repository, Documentation, Github actions ...)

1 Introduction

2 Finite Element Methods (FEMs)

3 Fourier Neural Operator (FNO)

We will now introduce Fourier Neural Operators (FNO). For more information, please refer to the following articles ADD REF !.
In image treatment, we call image tensors of size $ni \times nj \times nk$, where $ni \times nj$ corresponds to the image resolution and nk corresponds to its number of channels. For example, an RGB (Red Green Blue) image has $nk = 3$ channels. We choose here to present the FNO as an operator acting on discrete images. Reference articles present it in its continuous aspect, which is an interesting point of view. Indeed, it is thanks to this property that it can be trained/evaluated with images of different resolutions.
The FNO methodology creates a relationship between two spaces from a finite collection of observed input-output pairs. Est-ce que je gardes cette phrase ?
A rajouter : - implémentation effectuée/fournie par Vincent Vigon
- on ne trouvera pas ici de test de variation des paramètres (modes, width...)

3.1 Architecture of the FNO

The following figure (Figure 3.1) describes the FNO architecture in detail:

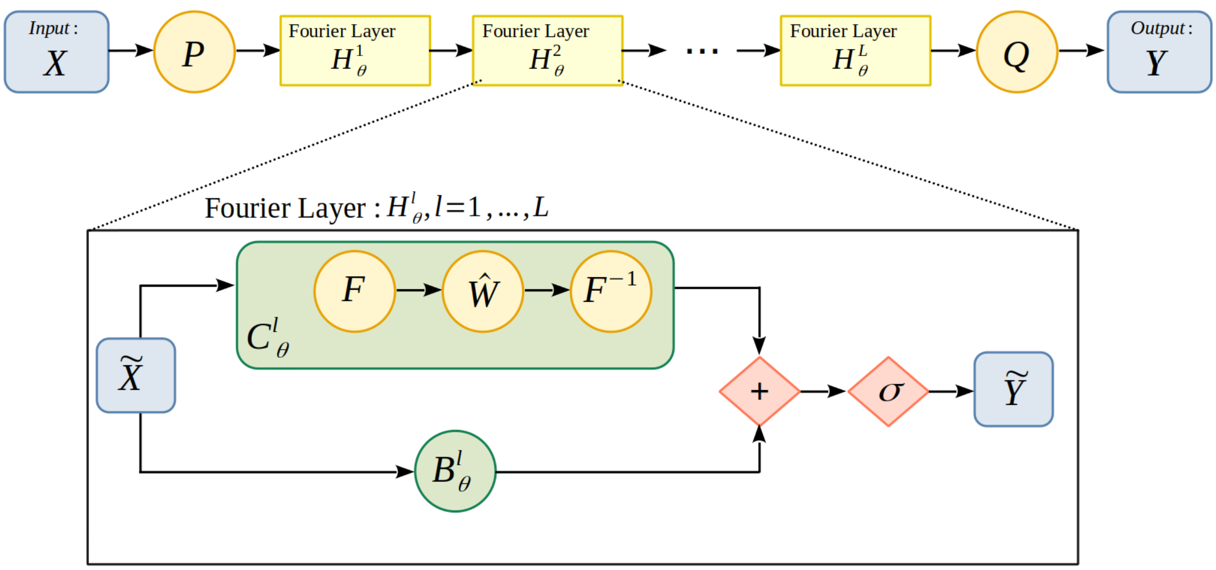

Figure 3.1: Architecture of the FNO

The architecture of the FNO is as follows:

$$G_\theta = P \circ \mathcal{H}^1_\theta \circ \cdots \circ \mathcal{H}^L_\theta \circ Q$$

3.2 General structure of the FNO

We'll now describe the composition of the Figure 3.1 in a little more detail :

- We start with input X of shape (batch_size, height, width, nb_channels) with batch_size the number of images to be processed at the same time, height and width the dimensions of the images and nb_channels the number of channels. Simplify by (bs,ni,nj,nk).
- We perform a P transformation in order to move to a space with more channels. This step enables the network to build a sufficiently rich representation of the data. For example, a Dense layer (also known as fully-connected) can be used.
- We then apply L Fourier layers, noted $\mathcal{H}_\theta^l$, $l = 1,\ldots,L$, whose specifications will be detailed in Section 3.3.
- We then return to the target dimension by performing a Q transformation. In our case, the number of output channels is 1.
- We then obtain the output of the Y model of shape (bs,ni,nj,1).

3.3 Fourier Layer structure

Each Fourier layer is divided into two sublayers:

$$\tilde{Y} = \mathcal{H}_\theta^l(\tilde{X}) = \sigma\left(\mathcal{C}_\theta^l(\tilde{X}) + \mathcal{B}_\theta^l(\tilde{X})\right)$$

where

- $\tilde{X}$ corresponds to the input of the current layer and $\tilde{Y}$ to the output.
- σ is an activation function. For $l = 1,\ldots,L-1$, we'll take the activation function ReLU (Rectified Linear Unit) and for $l = L$ we'll take the activation function GELU (Gaussian Error Linear Units).

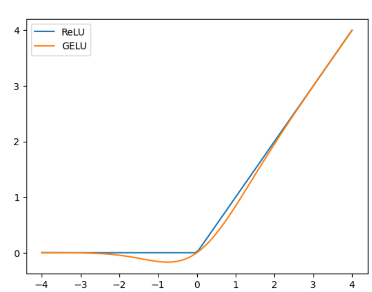

Figure 3.2: Activation functions used

- $\mathcal{C}_\theta^l$ is a convolution layer where convolution is performed by FFT (Fast Fourier Transform). For more details, see Section 3.3.1.
- $\mathcal{B}_\theta^l$ is the "bias-layer". For more details, see Section 3.3.2.

3.3.1 Convolution sublayer

Each $\mathscr{C}_\theta^l$ convolution layer contains a trainable kernel $\hat{W}$ and performs the transformation

$$\mathscr{C}_\theta^l(X) = \mathscr{F}^{-1}(\mathscr{F}(X)\cdot\hat{W})$$

where $\mathscr{F}$ corresponds to the 2D Discrete Fourier Transform (DFT) on a $ni \times nj$ resolution grid and

$$(Y\cdot\hat{W})_{ijk} = \sum_{k'} Y_{ijk'}\hat{W}_{ijk'}$$

In other words, this transormation is applied channel by channel.

Remark. *An image is fundamentally a signal. Just as 1D signals show changes in amplitude (sound) over time, 2D signals show variations in intensity (light) over space. The Fourier transform allows us to move from the spatial or temporal domain into the frequency domain. In a sound signal (1D signal), low frequencies represent low-pitched sounds and high frequencies represent high-pitched sounds. In the case of an image (2D signal), low frequencies represent large homogeneous surfaces and blurred parts, while high frequencies represent contours, more generally abrupt changes in intensity and, finally, noise.*

The 2D DFT is defined by :

$$\mathscr{F}(X)_{ijk} = \frac{1}{ni}\frac{1}{nj}\sum_{i'=0}^{ni-1}\sum_{j'=0}^{nj-1} X_{i'j'k}e^{-2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)}$$

The inverse of the 2D DFT is defined by :

$$\mathscr{F}^{-1}(X)_{ijk} = \sum_{i'=0}^{ni-1}\sum_{j'=0}^{nj-1} X_{i'j'k}e^{2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)}$$

We can easily show that $\mathscr{F}$ is the reciprocal function of $\mathscr{F}^{-1}$. We have

$$\begin{aligned}
\mathscr{F}^{-1}(\mathscr{F}(X))_{ijk} &= \sum_{i'=0}^{ni-1}\sum_{j'=0}^{nj-1}\mathscr{F}(X)_{i'j'k}e^{2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)}\\
&= \frac{1}{ni}\frac{1}{nj}\sum_{i'j'}\sum_{i''j''}X_{i''j''k}e^{-2\sqrt{-1}\pi\left(\frac{i'i''}{ni}+\frac{j'j''}{nj}\right)}e^{2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)}\\
&= \frac{1}{ni}\frac{1}{nj}\sum_{i''j''}X_{i''j''k}\underbrace{\sum_{i'j'}e^{2\sqrt{-1}\pi\frac{i'}{ni}(i-i'')}e^{2\sqrt{-1}\pi\frac{j'}{nj}(j-j'')}}_{:=S}
\end{aligned}$$

Thus

- If $(i,j)=(i'',j'') : S = \sum_{i',j'} 1 = ni\times nj$

- If $(i,j) \neq (i'',j'')$:

$$\begin{aligned} S &= \sum_{i'} \left(e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')} \right)^{i'} \sum_{j'} \left(e^{\frac{2\sqrt{-1}\pi}{nj}(j-j'')} \right)^{j'} \\ &= \frac{1-\left(e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')} \right)^{ni}}{1-e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')}} \times \frac{1-\left(e^{\frac{2\sqrt{-1}\pi}{nj}(j-j'')} \right)^{nj}}{1-e^{\frac{2\sqrt{-1}\pi}{nj}(j-j'')}} \\ &= \frac{1-e^{2\sqrt{-1}\pi(i-i'')}}{1-e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')}} \times \frac{1-e^{2\sqrt{-1}\pi(j-j'')}}{1-e^{\frac{2\sqrt{-1}\pi}{ni}(j-j'')}} = 0 \end{aligned}$$

because if $i \neq i''$

$$e^{2\sqrt{-1}\pi(i-i'')} = cos(2\pi(i-i'')) + \sqrt{-1}sin(2\pi(i-i'')) = 1+0 = 1$$

and if $j \neq j''$

$$e^{2\sqrt{-1}\pi(j-j'')} = cos(2\pi(j-j'')) + \sqrt{-1}sin(2\pi(j-j'')) = 1+0 = 1$$

We deduce that

$$\mathscr{F}^{-1}(\mathscr{F}(X))_{ijk} = \frac{1}{ni}\frac{1}{nj} \times ni \times nj \times X_{ijk} = X_{ijk}$$

And finally $\mathscr{F}$ is the reciprocal function of $\mathscr{F}^{-1}$.
For more details about the Convolution sublayer, see Section 3.4.

3.3.2 Bias subLayer

The bias layer is a 2D convolution with a kernel size of 1. This means that it only performs matrix multiplication on the channels, but pixel by pixel. In other words, it mixes channels via a kernel, but does not allow interaction between pixels.
Precisely,

$$\mathscr{B}^l_\theta(X)_{ijk} = \sum_{k'} X_{ijk} W_{k'k} + B_k$$

3.4 Some details on the convolution sublayer

In this section, we will specify some details for the convolution layer.

3.4.1 Border issues

Let $W = \mathscr{F}^{-1}(\hat{W})$, we have :

$$\mathscr{C}^l_\theta(\tilde{X}) = \mathscr{F}^{-1}\left(\mathscr{F}(X)\cdot\hat{W}\right) = \tilde{X} \star W$$

with

$$(\tilde{X} \star W)_{ij} = \sum_{i'j'} \tilde{X}_{i-i'[ni],j-j'[nj]} W_{i'j'}$$

In other words, multiplying in Fourier space is equivalent to performing a $\star$ circular convolution in real space. But these modulo operations are only natural for periodic images, which is not our case. A compléter (expliquer padding par ex)

3.4.2 FFT

To speed up computations, we will use the FFT (Fast Fourier Transform). The FFT is a fast algorithm to compute the DFT. It is recursive : The transformation of a signal of size N is make from the decomposition of two sub-signals of size $N/2$. The complexity of the FFT is $N\log(N)$ whereas the natural algorithm, which is a matrix multiplication, has a complexity of N^2.

3.4.3 Real DFT

In reality, we'll be using a specific implementation of FFT, called RFFT (Real Fast Fourier Transorm).In fact, for $\mathscr{F}^{-1}(A)$ to be real if A is a complex-valued matrix, it is necessary that A respects the Hermitian symmetry:

$$A_{i,nj-(j+1)} = \bar{A}_{i,j}$$

In our case, we want $\mathscr{C}^l_\theta(X)$ to be a real image, so $\mathscr{F}(X)\cdot\hat{W}$ must verify Hermitian-symmetry. To do this, we only need to collect half of the Discrete Fourier Coefficients (DFC) and the other half will be deduced by Hermitian symmetry. More precisely, using the specific RFFT implementation, the DFCs are stored in a matrix of size $(ni, nj//2+1)$. Multiplication can then be performed by the $\hat{W}$ kernel, and when the inverse RFFT is performed, the DFCs will be automatically symmetrized. So the Hermitian symmetry of $\mathscr{F}(X)\cdot\hat{W}$ is verified and $\mathscr{C}^l_\theta(X)$ is indeed a real image.
To simplify, let's assume nk=1. Here is a diagram describing this idea:

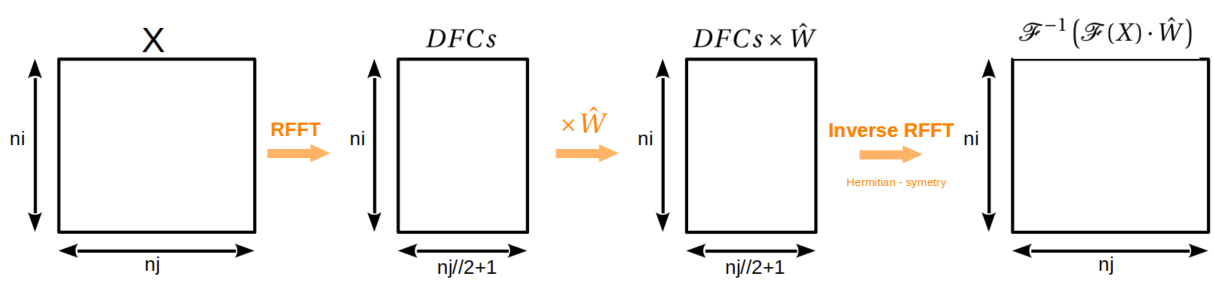

Figure 3.3: RFFT with Hermitian-symmetry scheme

3.4.4 Low pass filter

When we perform a DFT on an image, the DFCs related to high frequencies are in practice very low. This is why we can easily filter an image by ignoring these high frequencies, i.e. by truncating the high Fourier modes. In fact, eliminating the higher Fourier modes enables a kind of regularization that helps the generalization. So, in practice, it's sufficient to keep only the DFCs corresponding to low frequencies. Typically, for images of resolution 32×32 to 128×128, we can keep only the 20×20 DFCs associated to low frequencies.
Here is a representation of this idea in 1D :

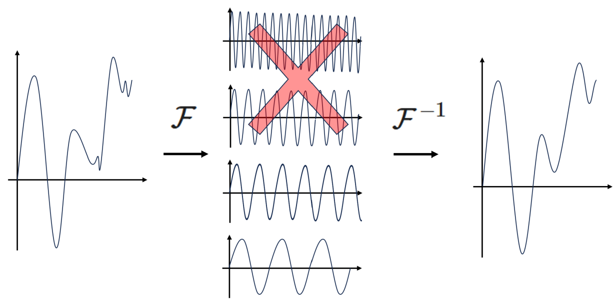

Figure 3.4: Low pass filter

3.4.5 Global aspect of the FNO

Classical Convolutional Neural Networks (CNN) use very small kernels (typically 3×3). This operation only has a local effect, and it's the sequence of many convolutions that produces more global effects.
In addition, CNNs often use max or mean-pooling layers, which process the image on several scales. Max-pooling (respectively mean-pooling) consists in slicing the image into small pieces of size $n \times n$, then choosing the pixel with the highest value (respectively the average of the pixels) in each of the small pieces. In most cases, $n = 2$ is used, which divides the number of pixels by 4.
The FNO, on the other hand, uses a $\hat{W}$ frequency kernel and $W = \mathscr{F}^{-1}(\hat{W})$ has full support. For this reason, the effect is immediately non-local. As a result, we can use less layers and we don't need to use a pooling layer.

3.5 Application

Dans notre cas, on souhaite apprendre au FNO à prédire des solutions d'EDP. Plus précisément, on souhaite que le réseau soit capable de prédire la solution à partir d'un terme source f. enrichissement des données, grilles régulières

4 Correction

5 Numerical Results ?

6 Conclusion