POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Music and Acoustic Engineering

Data Driven Methods for Frequency Response Functions Interpolation

by:
Matteo Acerbi

matr.:
921328

Supervisor:
Fabio Antonacci

Co-supervisor:
Raffaele Malvermi

Academic Year
2019-2020

POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Music and Acoustic Engineering

Metodi data-driven per interpolazione di dati vibrometrici

Candidato:
Matteo Acerbi

matr.:
921328

Relatore:
Fabio Antonacci

Co-relatore:
Raffaele Malvermi

Anno Accademico
2019-2020

Abstract

In the field of structural mechanics, classical methods for the vibrational characterization of objects exploit the inherent redundancy of a relevant amount of measurements acquired over regular sampling grids. However, there are cases in which parts of the objects under analysis are not accessible with sensors, leading to irregular sampling grids characterized by holes. These problems inevitably concern also specific studies in the field of Musical Acoustics, where vibrational analyses conducted on musical instruments or parts of them, such as violins top plates, reveal the impossibility of acquiring data at any point on the instrument surface. Interpolation methods could help in overcoming this issue, by retrieving missing information on the basis of the one available. Model-based interpolation methods are the most popularly used in commercial software, based on the usage of prior knowledge, most of the time expressed in terms polynomial functions, to interpolate data. Those methods have proved to be effective if applied on regular down-sampled grids, but their performances actually really depends on the amount of data available and could particularly get worse if applied on irregular grids, thus failing in real experimental contexts such as vibration data acquisitions.

In this thesis we propose to use Convolutional Autoencoders (CA) for Frequency Response Function (FRF) interpolation over grids with different subsampling schemes. CA learn a compressed representation from a dataset of FRFs synthetized through Finite Element Analysis. Tests with numerical and experimental data show the effectiveness of the model with a different amount of missing data and its ability to predict real FRFs characterized by different damping and sampling frequency.

Sommario

Nel campo della meccanica strutturale, i metodi classici per la caratterizzazione vibrazionale dei corpi sfruttano la ridondanza delle misurazioni acquisite su griglie di campionamento regolari. Tuttavia, vi sono casi in cui parti dei sistemi sottoposti a misurazione non sono accessibili mediante sensori, il che porta ad avere griglie di campionamento irregolari, spesso caratterizzate dalla presenza di fori e dati mancanti. Queste problematiche riguardano inevitabilmente anche specifici studi nel settore dell'Acustica Musicale, dove le analisi vibratorie condotte su strumenti musicali o parti di essi, come le tavole armoniche dei violini, rivelano l'impossibilita di acquisire dati in un qualsivoglia punto della loro superficie. Algoritmi di interpolazione possono essere uno stumento efficace al fine di contravvenire a tal problema, recuperando informazione mancante sulla base di quella acquisita. La categoria più comune è quella dei metodi di interpolazione model-based, per altro i più popolari nei software ad uso commerciale. Essi si basano sull'utilizzo di assunzioni a priori, la maggior parte delle volte espresse in termini di funzioni polinomiali, al fine di performare uno schema di interpolazione. Tali metodi si dimostrano efficaci se applicati su griglie sottocampionate regolarmente, ma d'altra parte le loro prestazioni risultano fortemente dipendenti dalla quantità di dati disponibili e possono notevolmente peggiorare se applicati su griglie di sottocampionamento irregolari, venendo meno in reali contesti sperimentali come acquisizioni di dati vibratori.

In questa tesi proponiamo l'utilizzo di Autoencoder Convoluzionali (CA) per l'interpolazione di Risposte in Frequenza (FRF) a partire da griglie afflitte da diversi schemi di sottocampionamento. Questa classe di architetture è in grado di imparare una rappresentazione compressa da un set di FRF sintetizzate mediante Finite Element Analysis (FEA). Test condotti su dati numerici e sperimentali mostrano l'efficacia del modello da noi proposto nel ricostruire strutture dati con una diversa quantità di dati mancanti e la sua capacità di predire FRF reali, indipendentemente dalla frequenza di campionamento utilizzata e dai fattori di smorzamento propri della struttura in analisi.

Acknowledgments

This thesis is the result of almost a year of work at the Image and Sound Processing Lab. First I would like to thank my supervisor Fabio Antonacci that gave me the opportunity to work on this fascinating topic. This allowed me to have a first insight on the world of research and to understand what actually means to work with a team in order to reach a common goal.

Thanks to my co-supervisor Raffaele Malvermi, for his patience and for the constant support he gave me to perform my studies. I am very grateful for his indefatigable supervision. It was determinant for me to reach the results reported in this thesis.

Last but not least, I want to thank my family and my parents in particular. They always believed in me and in my aspirations, giving me the possibility to follow my own way that for now, but only for now, ends up with this thesis, which I see as the result of their faith. They gave me the chance to be a free man. With the lovely aid they constantly provided to me, my work became easier to pursuit during such a year full of difficulties.

M.A.

Contents

List of Figures

List of Tables

Introduction

In the field of Musical Acoustics, the vibration modes of wooden music instruments like violins provide insights on the dynamics of the instrument itself. Indeed, modal parameters, i.e. natural frequencies and damping ratios, as well as deformation patterns (mode shapes) associated with them are known to have a large effect on the sound produced. For example, a different placement of the bassbar and the lack of arching in a trapezoidal violin can result in more symmetric vibration modes, which on their turn result in a lack of brilliance in the sound.

A traditional method that luthiers use to detect the modes of violins top plate is the Chaldni method [1]. The idea is to put some sand on the plate and use a shaker to excite the structure over a range of frequencies. When the driving frequency fits one natural frequency of the plate, the mode at that frequency is excited, and the sand will collect near the nodal lines characterizing that mode. By applying this method, luthiers can easily determine the frequencies of the normal modes as well as having a rough estimation of the mode shapes. Nowadays, Experimental Modal Analysis (EMA) is the most common technique that aims to extract the modal parameters characterizing the vibrating structure of interest, starting from Frequency Response Function (FRF) measurements obtained through vibration tests performed on the structure itself. This task is usually performed not only for the characterization of musical instruments [2], but in a wide range of applications, e.g. the monitoring of civil structures [3]. For what concerns specifically the design of musical instruments like violins, since there is a strong relation between vibrational modes and the actual sound radiation, EMA can explain differences in the radiated sound by extracting the instrument modal parameters. As such, EMA is a valuable tool to examine the performance of experimental music instruments, supporting luthiers during the design phase.

Classical methods for EMA exploit the inherent redundancy of measurements in time or frequency domain [4, 5, 6]. It follows that the ability of describing the vibrational behavior of a continuous system directly depends on the finite set of spatial points analyzed. However, there are cases in which parts of the objects under analysis are not accessible with sensors. For example, in the case of violins, a section of the top plate is covered by the fingerboard and its response has to be predicted starting from the measurements acquired over the remaining surface. These particular scenarios let EMA dealing with irregular sampling grids characterized by irregularly spaced points, where classical interpolation schemes offer fast estimations but often fail in filling the gaps with meaningful values.

A wide range of interpolation algorithms is offered by the literature. Fourier-based interpolation methods, for example, perform interpolation by calculating the input data spectrum, applying zero padding and finally coming back to the original domain to obtain the corresponding super-resolved data. Bicubic interpolation (BCI) is actually the most used interpolation method in commercial software, and it is based on the use of prior cubic polynomials to lead the generation of new data.

Recent works have tackled the problem of irregularly sampled data, contextualized to the specific field of vibration studies. Chardon *et al.* [7] used Compressed Sensing to synthesize plate impulse responses from a coarser sampling grid as a sparse combination of plane waves. The method works for any star-convex geometry and boundary condition, but still needs a regular grid as starting point. Schwarz *et al.* [8] were able to generate signals over a finer grid of points by decomposing the available measurements into a summation of resonance curves and fitting them with Finite Element Analysis (FEA) mode shapes. In this case, the method can work also with irregular sets of FRFs but requires a specific Finite Element Model (FEM) of the object to be set.

In image processing, Artificial Neural Networks (ANNs) have become the state-of-the-art solution for similar restoration tasks such as super-resolution [9, 10, 11, 12, 13], denoising and inpainting [14, 15], also when dealing with scientific data [16, 17, 18, 19]. The results achieved are promising, as those methods have shown to be effective not only when reconstructing images for which a perceptive accuracy was needed, but also when the specific information contained in the data needed to be retrieved. Mandelli *et al.* [16], in particular, proposed a method for the reconstruction of corrupted seismic data, based on the use of Convolutional Neural Networks (CNNs). Their purpose was specifically to perform interpolation and denoising of 2D shot gathers. Results achieved on controlled synthetic experiments demonstrated that the proposed method is a promising strategy for seismic data pre-processing. This method indeed is capable to efficiently restore 2D corrupted data, and also to deal with the task of spatially upsampling the shot gathers. Moreover, once the network training procedure was completed, processing data with their strategy resulted to be also efficient in terms of computational effort.

Inspired by the first results obtained in [20] on mode shape super-resolution with CNNs and in [21] on noise transfer function estimation with Feed-Forward Neural Networks, we propose to investigate further in this direction by considering Convolutional Autoencoders (CA) trained on FRFs simulated through a FEM model of a thin rectangular plate. The adoption of this architecture is due to its ability of learning a compressed representation from data by design, thanks to its encoding stage. In particular, we implement here the renowned U-net, introduced for the first time in [22] for segmentation of medical images.

At first, we defined a U-net architecture dealing with 2D space-frequency images, where FRF data are stored as a function of frequency f and one plate spatial dimension. We did so since the most common procedure proposed by the literature when applying deep learning models to image processing tasks is to deal with 2D images. Then, we decided to move on the definition of a U-net architecture dealing with 3D FRF tensors, where FRF data are stored as a function of frequency f and both 2D spatial coordinates (x, y). This choice has been made as in our work FRF data are acquired over sampled points disposed on a 2D plate surface, so the 3D U-net architecture was defined with the purpose of having a model able to combine the information along both the system spatial axes, overcoming the intrinsic limit of the 2D U-net that can manage FRF data along only one spatial dimension.

Preliminary frequency domain studies have been conducted on a FEM model of a thin plate defined thorugh COMSOL Multiphysics®, obtaining synthetic FRF measurements and building the dataset used for training, validation and testing of both the 2D and the 3D U-nets. To avoid the networks to get stuck in local minima due to the sparse nature of the data considered, we defined a modified version of the Mean Square Error loss function where a mask concentrates the attention of the interpolator on specific features characterizing the FRF shapes, i.e. resonances, antiresonances and boundary conditions. Finally, we applied the 2D U-net to FRF measurements acquired from a real rectangular plate, to see the performance of a model trained on synthetic data if applied on real ones.

Experiments with synthetic and real measured data show the effectiveness of the models compared to classical spatial interpolation techniques and their ability to predict real FRFs characterized by different damping and sampling frequency with respect to the training data.

The thesis is organized as follows. In Chapter 1 we give a background on vibration analysis, starting from considering the simplest case of the mechanical harmonic oscillator, by then moving on the study of continuous systems such as thin plates, exploring both free and forced vibration, to finally introduce the concept of FRF, comparing EMA with FEA to retrieve modal parameters from FRF data. The problem of irregularly sampled data in the field of vibration studies will be exposed, and an introduction to data interpolation as a solution to this problem will be given, offering an overview on the state of the art available in the field of super-resolution algorithms. In particular, applications of Deep Learning to solve such inverse problems will be considered. A background on ANNs will thus be given, to finally describe the architecture proposed in this work, i.e. the U-net.

In Chapter 2 we firstly offer a mathematical formulation of the problem, seeing how FRFs measurements obtained from a rectangular thin plate can be stored either in terms of 3D tensors, representing data as a function 2D space and frequency, or in terms of 2D space-frequency images, obtained by slicing those tensors along one spatial dimension. We will thus introduce two U-net architectures, the first one dealing with 2D space-frequency images and the second one with 3D tensors, both with the purpose of interpolating FRF signals. A specific section will be dedicated to the definition of the custom loss function used to train the models.

In Chapter 3 it will be described all the steps followed in order to build the datasets needed to train, validate and test our models, starting from the description of the simulations setup in a COMSOL Multiphysics® environment, up to the use of Python to define the final data structures into which organize the synthetic FRFs.

In Chapter 4 we present the main results of the tests performed to validate our models. Results concerning 2D space-frequency images interpolation will be shown at first, by then moving on the analysis of 3D FRF tensors. In both the cases, it will be considered initially the more ideal case of interpolation on regular grids, then moving to the harder and more realistic case of interpolating FRF measurements on irregular grids.

Finally, Chapter 5 draws the conclusions of the work and gives some insights on the future work and applications.

1 Background and State of the Art

In this chapter we will introduce the reader to the main concepts behind vibration analysis. In order to do so, we will start dealing with the simplest vibratory system that can be considered, i.e. the damped harmonic oscillator, moving then to continuous systems like thin plates in order to concentrate the analysis on structures that can model musical instruments parts, such as violin plates, and their behaviour. Both free and forced vibration will be analyzed, introducing to the concepts of natural frequency, mode shape and FRF, starting by dealing with lumped systems with only one degree of freedom (DOF), up to continuous systems with a possibly infinite number of DOFs as violin plates. The discussion will then move on how FRF data can be retrieved to perform modal analysis, either experimentally through EMA or numerically through FEA.

The subsequent section will instead be devoted to the problem of irregularly sampled FRF data and on how super-resolution algorithms can help in order to perform data interpolation, retrieving meaningful missing information. The state of the art on super-resolution will thus be examined, exploring the different methodologies adopted up to now and introducing ANNs as a promising class of algorithms to perform super-resolution.

A background on Machine Learning will thus be given, by then moving on the treatment of ANNs with a focus on CNNs, finally introducing and describing the architecture adopted by us in this work, i.e. the U-net.

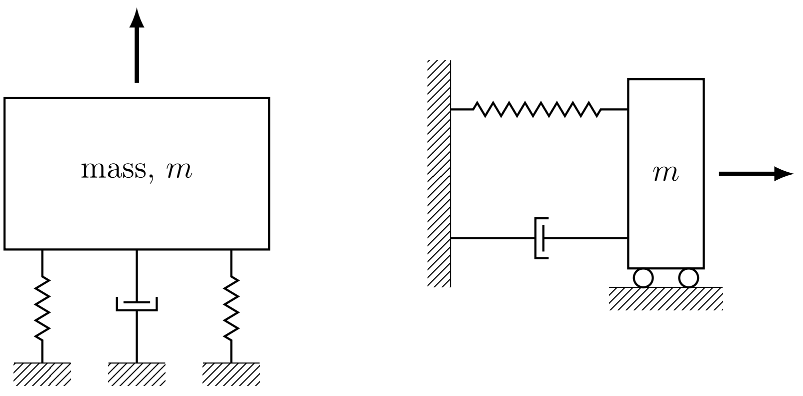


Figure 1.1: Two examples of damped harmonic oscillator

1.1 Background on Vibration Analysis

1.1.1 Simple Mechanical Oscillator

In our life there are many examples of oscillation. Some of them are the vibrating strings of a picked guitar, the vibrating vocal cords while people are speaking and the pendulum of a clock swinging from right to left. Oscillations can occur when a system is disturbed from its stable equilibrium position, leading to a characteristic periodic motion as response. Intuitively speaking, vibration can be described as the back and forth motion of a system, which can be continuous, regular and repetitive but can also irregular or randomic.

Vibration Analysis is defined as the study of mechanical structures vibrations and when also acoustical effects are accounted for, we talk about **Vibroacoustics**. The latter is particularly interesting when applied on musical instruments. Numerous studies have been performed on violins plates [23, 24, 25, 26], where vibration analysis help in defining the dynamic properties describing those structures, helping also in understanding the relation between those properties and the quality of the sound produced.

Before analyzing the vibratory behaviour of continuous systems such as a violin plate, it is necessary to give first an introduction on vibration theory, by treating the simplest mechanical system possible, i.e. the **damped harmonic oscillator** [27]. A schematic representation of it is given in Fig. 1.1. This system can be modelled as a connection of three blocks:

- A **body** with mass m;
- A **spring** with stiffness k (or compliance $C = \frac{1}{k}$, conversely);
- A **damper** with damping c.

Let us first consider the **free vibration** of the system, i.e. the system response when it is let be free to oscillate after having imposed some initial conditions that differ from its equilibrium state. In this case, the forces acting on the body with mass m are:

- the **inertia force** $F_m = m\ddot{x}$ associated to the mass m of the body ($\ddot{x}$ is the acceleration of the body itself)
- the **elastic force** $F_{el} = kx$ associated to the spring, given by the Hooke's law (x represents the displacement of the body with respect to its rest position.)
- the **friction force** $F_d = c\dot{x}$ associated to the damper ($\dot{x}$ is the velocity associated to the body)

The balance of forces (Newton's second law) for a damped harmonic oscillator gives thus the following equation of motion

$$m\ddot{x} + c\dot{x} + kx = 0, \tag{1.1}$$

By defining $\omega_0 = \sqrt{\frac{k}{m}}$ as the **natural frequency** of the undamped system and $\alpha = \frac{c}{2m}$, (1.1) can be rewritten as

$$\ddot{x} + 2\alpha\dot{x} + {\omega_0}^2 x = 0. \tag{1.2}$$

A general solution for (1.2) should have the form $\tilde{x} = \tilde{A}e^{\gamma t}$, and substituting it into the equation of motion yelds

$$(\gamma^2 + 2\alpha\gamma + {\omega_0}^2)\tilde{A}e^{\gamma t} = 0, \tag{1.3}$$

which implies $\gamma^2 + 2\alpha\gamma + {\omega_0}^2 = 0$ as unique non trivial solution, i.e.

$$\gamma = -\alpha \pm \sqrt{\alpha^2 - {\omega_0}^2} = -\alpha \pm j\sqrt{{\omega_0}^2 - \alpha^2} = -\alpha \pm j\omega_d. \tag{1.4}$$

The general solution therefore becomes

$$\tilde{x} = e^{-\alpha t}(\tilde{A}_1 e^{j\omega_d t} + \tilde{A}_2 e^{-j\omega_d t}), \tag{1.5}$$

where expressions of $\tilde{A}_1$ and $\tilde{A}_2$ can be obtained in terms of the initial velocity and acceleration applied on the system. The system response can thus be described as a complex sinusoid with carrier frequency ω_d, whose amplitude decreases over time due to the presence of the $e^{-\alpha t}$ term accounting for the damping effects on the system itself. It is important to notice that $\omega_d = \sqrt{\omega_0^2 - \alpha^2} \neq 0$ which implies that the natural frequency of a damped system is different from that of a conservative system with the same mass and spring. In mechanical and acoustic systems of our interest, however, we can always approximate $\omega_d = \omega_0$, as α is very small. When $\alpha > \omega_0$, the exponent becomes real and the system is not anymore oscillatory but instead exhibits an exponential attenuation. In Fig. 1.2 is shown the system free vibration plotted over time, where it can be noted the amplitude modulation to which the sinusoidal displacement is subjected as α increases. In real applications, what is interesting to

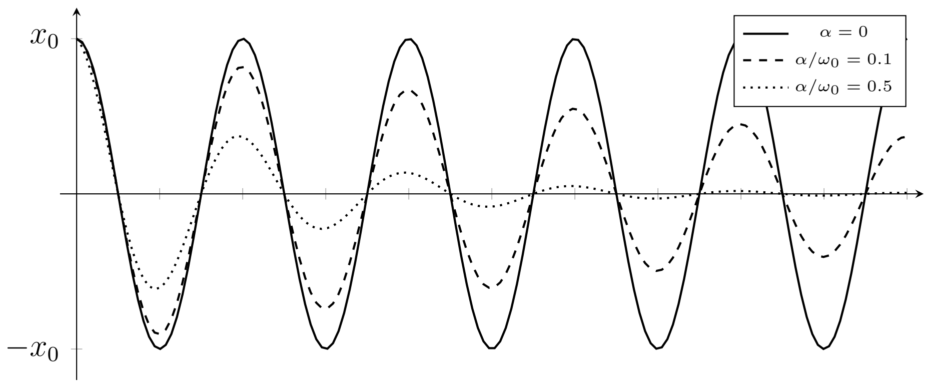


Figure 1.2: Harmonic vibration over time, assuming initial velocity $v_0 = 0$, and considering different values of α.

study is the **forced vibration** of a system, i.e. when the system itself is subjected to force constantly acting on it. Let us consider a sinusoidal force $\tilde{F} = Fe^{j\omega t}$ applied at time $t = 0$ to a damped oscillator. The equation of motion will thus become

$$m\ddot{x} + c\dot{x} + kx = Fe^{j\omega t}. \tag{1.6}$$

Since the system is linear, its **steady-state response** will be at the same frequency of the driving force. The general harmonic solution will be $\tilde{A}e^{j\omega t}$ so, evaluating (1.6) in the frequency domain and solving for $\tilde{X}(\omega)$, we obtain

$$\tilde{X} = \frac{Fe^{j\omega t}}{k - \omega^2 m + j\omega c} = \frac{\tilde{F}/m}{\omega_0^2 - \omega^2 m + j\omega 2\alpha}, \tag{1.7}$$

from which we can compute the **receptance** H as

$$H = \frac{\tilde{x}}{\tilde{F}} = \frac{1/m}{\omega_0^2 - \omega^2 m + j\omega 2\alpha}. \tag{1.8}$$

This indicator describes the system response, in terms of displacement, as a function of the load frequency. What happens is that the receptance H will have a maximum in magnitude (resonance) in correspondence of $\omega = \omega_0$, meaning that if an harmonic load with carrier frequency ω_0 would be applied on the system under analysis, the system itself would respond oscillating with the maximum displacement possible. We will retrieve this concept in the study of continuous systems in Sec. 1.1.3.

Another important quantity is the so called **mobility** Y, which describes the system response as a function of the load frequency, but this time in terms of velocity. Since velocity is the derivative with respect to time of the displacement, and since time derivative corresponds to a product by a factor of $j\omega$ in the frequency domain, mobility can easily be derived from receptance as

$$Y = j\omega \cdot H. \tag{1.9}$$

The response of a system can be described also in terms of acceleration, and in this case we talk of **accelerance** A, which in the frequency domain can be derived as further multiplying the mobility by $j\omega$, namely

$$A = j\omega \cdot Y = -\omega^2 \cdot H. \tag{1.10}$$

Usually it is also interesting to evaluate another metric, which is the **impedance**, that can be derived as the reciprocal of the mobility. It relates the system response in terms of velocity to the load frequency, namely:

$$Z = \frac{1}{Y} = \frac{\tilde{F}}{\tilde{v}} = c + j\left(\omega m - \frac{k}{\omega}\right), \tag{1.11}$$

where the real part is usually denoted with R, to indicate a mechanical resistance, so the above equation becomes:

$$Z = R + jX, \tag{1.12}$$

where $X = \omega m - \frac{k}{\omega}$ is called mechanical reactance.

Receptance, mobility and accelerance are output/input relations, as the system output, either in terms of displacement or velocity or acceleration, is analyzed as a function of the input load, while, on the other side, the impedance is an input/output relation and what actually happens is that, in correspondence of a natural frequency, this last metric will present a minimum of magnitude. However, all these metrics offer different representations of the same phenomenon and are strictly correlated one to the other. They are indeed all FRFs, because of relating system output and input. In this work, the system output (the plate response to harmonic excitation) will be retrieved in terms of displacement, so we will manage receptance signals, referring to them as FRFs only for simplicity.

It is usually helpful, in order to perform comparative analisys, to plot both the impedance and the mobility, also called **admittance**, over frequency as in Fig. 1.3.

The mechanical oscillator on which we have focused our attention since now is actually a simple mechanical system that can be treated as a lumped model having only one DOF, i.e. the displacement x associated to the translational motion of the body with mass m. We have deeply analyzed the behaviour of such a system, both in free and forced conditions, underlying the presence of a single natural frequency ω_d completely determined by the body mass m, the spring stiffness k and the damping c. However, real structures as for example musical instruments, are constituted of a variety of elements, such as plates, membranes or tubes, and they must be studied as continuous systems, whose motion is described by a more complex set of differential equations.

In general, a continuous system can be treated as a n DOFs system, with $n \to \infty$. The equations describing its motion are expressed through partial derivatives, since the motion of the system depends both on time

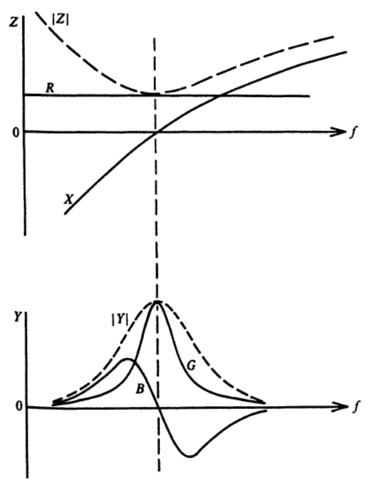


Figure 1.3: Impedance and Admittance plotted over frequency. On the top the impedance, on the bottom the admittance.

and on one or more spatial coordinates. In particular, objects like strings and bars can be considered as one-dimensional continuous systems, since one spatial dimension is much greater than the other two, while in case of thin plates and membranes, only the thickness is negligible with respect to width and length, and therefore they are suitable to be modeled as two-dimensional continuous systems.

1.1.2 Vibration analysis of rectangular thin plates

In our work we are interested on specific continuous systems which are violin plates. As already mentioned, in a preliminary analysis thin plates can be treated as 2D continuous systems, where thickness can be neglected. In this section we thus aim at giving a background on free vibration of thin plates, showing how the concepts introduced in the section before for the case of a simple mechanical oscillator, are still valid and can be generalized to continuous systems such as violin plates. We will analyze in deep plate vibration under simply supported boundary conditions, that is actually the only case possible for which it can be found an analytical solution of the problem in a closed form, giving us the possibility to have insights into plate response.

We start by introducing the Kirchhoff-Love assumptions [28] taken to perform the analysis:

- the plate is flat with constant thickness s;
- the plate is thin, i.e. its thickness s is much smaller than the other two dimensions (therefore, the plate can be regarded to as a 2D continuous system);
- homogeneous, isotropic and linear elastic material;

- small transverse displacements w orthogonal to the plate surface;
- a line segment initially normal to the undeformed xy mid plane remains straight and normal to the elastic surface under bending; therefore the displacements u and v (in the x and y directions) are proportional to the distance z from the mid plane;
- shear deformations are neglected;
- the normal stress σ_z is neglected.

According to the assumptions listed above, the **plate equation for bending vibration** is defined as

$$\nabla^4 w = -\frac{\rho s}{B}\frac{\partial^2 w}{\partial t^2}, \tag{1.13}$$

being $w = w(x, y, t)$ the surface displacement (along the z direction), for a plate having density ρ, thickness s, and bending stiffness B defined as

$$B = \frac{E}{1-\nu^2}\frac{s^3}{12}, \tag{1.14}$$

where E is the Young's modulus and ν is the Poisson's ratio, both associated to the specific plate material. The operator ∇^4 can be decomposed as

$$\nabla^4 = \nabla^2(\nabla^2 w), \tag{1.15}$$

where ∇^2 is the Laplace operator, expressed in Cartesian coordinates as

$$\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right), \tag{1.16}$$

so, in Cartesian coordinates, eq. 1.13 becomes

$$\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} = -\frac{\rho s}{B}\frac{\partial^2 w}{\partial t^2}. \tag{1.17}$$

We can impose a **standing wave solution**, defined as:

$$w(x, y, t) = \Phi(x, y) \cdot G(t). \tag{1.18}$$

The standing wave solution represents synchronous motion, that is a motion in which the shape of the plate displacement does not change with time, while the amplitude of this shape does: the ratio between the plate displacement evaluated in two fixed positions on the plate is constant at any time. (1.17) thus can be rearranged in order to split the spatial and the temporal contributions, leading to

$$\frac{B}{\rho s}\left(\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4}\right)\frac{1}{\Phi} = -\frac{1}{G}\frac{d^2 G}{dt^2}. \tag{1.19}$$

For the equation above to be satisfied, both members need to be equal to the same constant. Moreover, since the system under study is being perturbed around its rest position, then the constant itself needs to be real positive. We chose the constant to be equal to ω^2, leading to

$$\ddot{G} + \omega^2 G = 0 \rightarrow G(t) = Ge^{j\omega t}, \tag{1.20}$$

$$\left(\frac{\partial^4 \Phi}{\partial x^4} + 2\frac{\partial^4 \Phi}{\partial x^2 \partial y^2} + \frac{\partial^4 \Phi}{\partial y^4}\right) - \frac{\rho s \omega^2}{B}\Phi = 0. \tag{1.21}$$

The system response is strictly dependent on its **boundary conditions**, affecting the spatial term $\Phi(x, y)$. Given a plate having width length a and height length b, and defined the two spatial coordinates $x \in [0, a]$ and $y \in [0, b]$, we can have:

- **Free edges**
  The plate is completely free to vibrate also in correspondence of the edges, where displacement is thus different from zero

$$w(0, y) = w(a, y) = w(x, 0) = w(x, b) \neq 0 \tag{1.22}$$

- **Simply supported edges**
  The plate is pinned to its supports in correspondence of the edges, where the plate displacement is thus zero but the rotation is free and no bending moments are experienced

$$\begin{cases} w(0, y) = w(a, y) = 0 \\ \frac{\partial^2 w}{\partial x^2}\Big|_{x=0} = \frac{\partial^2 w}{\partial x^2}\Big|_{x=a} = 0 \\ w(x, 0) = w(x, b) = 0 \\ \frac{\partial^2 w}{\partial y^2}\Big|_{y=0} = \frac{\partial^2 w}{\partial y^2}\Big|_{y=b} = 0 \end{cases} \tag{1.23}$$

- **Clamped edges**
  Both displacement and rotation of the plate edges are prevented

$$\begin{cases} w(0, y) = w(a, y) = 0 \\ \frac{\partial w}{\partial x}\Big|_{x=0} = \frac{\partial w}{\partial x}\Big|_{x=a} = 0 \\ w(x, 0) = w(x, b) = 0 \\ \frac{\partial w}{\partial y}\Big|_{y=0} = \frac{\partial w}{\partial y}\Big|_{y=b} = 0 \end{cases} \tag{1.24}$$

As already said at the beginning of this section, we are going to consider the case of simply supported edges. We first impose the 4 boundary conditions on the two supported edges parallel to the y axis, namely

$$\begin{cases} w(0, y, t) \rightarrow \Phi(0, y) = 0 \\ w(a, y, t) \rightarrow \Phi(a, y) = 0 \\ m_y(0, y, t) = 0 \rightarrow \frac{\partial^2 \Phi}{\partial x^2}\Big|_{x=0} = 0 \\ m_y(a, y, t) = 0 \rightarrow \frac{\partial^2 \Phi}{\partial x^2}\Big|_{x=a} = 0 \end{cases} . \tag{1.25}$$

where $m_y(\cdot)$ indicates the bending moment along the y axis. A function Φ satisfying the 4 boundary conditions, is

$$\Phi(x, y) = F(y) sin\left(\frac{m\pi}{a}x\right), \qquad (n = 1, 2, ...) \tag{1.26}$$

and substituting it into (1.21), the following 4th order ordinary homogeneous differential equation in $F(y)$ is obtained as

$$\frac{d^4F}{dy^4} - 2\left(\frac{m\pi}{a}\right)^2 \frac{d^2F}{dy^2} + \left[\left(\frac{d^4F}{dy^4}\right)^4 - \frac{\rho s\omega^2}{B}\right] F = 0. \tag{1.27}$$

Imposing the characteristic solution yields

$$\lambda^4 - 2\left(\frac{m\pi}{a}\right)^2 \lambda^2 + \left[\left(\frac{m\pi}{a}\right)^4 - \frac{\rho s\omega^2}{B}\right] = 0, \tag{1.28}$$

and solving in λ we find

$$\lambda^2 = \left(\frac{m\pi}{a}\right)^2 \pm \sqrt{\frac{\rho s\omega^2}{B}} = \left(\frac{m\pi}{a}\right)^2 (1 \pm q), \tag{1.29}$$

where

$$\lambda_{1,2} = \pm\gamma_1 \rightarrow \gamma_1 = \frac{m\pi}{a}\sqrt{q+1}, \tag{1.30a}$$

$$\lambda_{3,4} = \pm\gamma_2 \rightarrow \gamma_2 = \frac{m\pi}{a}\sqrt{q-1}. \tag{1.30b}$$

The following solution in $F(y)$ is thus obtained as

$$F(y) = A\sin(\gamma_2 y) + B\cos(\gamma_2 y) + C\sinh(\gamma_1 y) + D\cosh(\gamma_1 y). \tag{1.31}$$

The unknowns A, B, C, D are computed by imposing the remaining 4 boundary conditions on the other two edges of the plate, i.e. those parallel to the x axis

$$\begin{cases} w(x, 0, t) \rightarrow \Phi(x, 0) = 0 \rightarrow F(0) = 0 \\ w(x, b, t) \rightarrow \Phi(x, b) = 0 \rightarrow F(b) = 0 \\ m_x(x, 0, t) = 0 \rightarrow \left.\frac{\partial^2\Phi}{\partial y^2}\right|_{y=0} = 0 \rightarrow F\text{"}(0) = 0 \\ m_x(x, b, t) = 0 \rightarrow \left.\frac{\partial^2\Phi}{\partial y^2}\right|_{y=b} = 0 \rightarrow F\text{"}(b) = 0 \end{cases} . \tag{1.32}$$

where $m_y(\cdot)$ indicates the bending moment along the y axis. The first two equations result into $B = D = 0$, while the other two can be reformulated according to the matrix equation

$$\begin{bmatrix} \sin(\gamma_2 b) & \sinh(\gamma_1 b) \\ -\gamma_2^2 \sin(\gamma_2 b) & \gamma_1^2 \sinh(\gamma_1 b) \end{bmatrix} \begin{bmatrix} A \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{1.33}$$

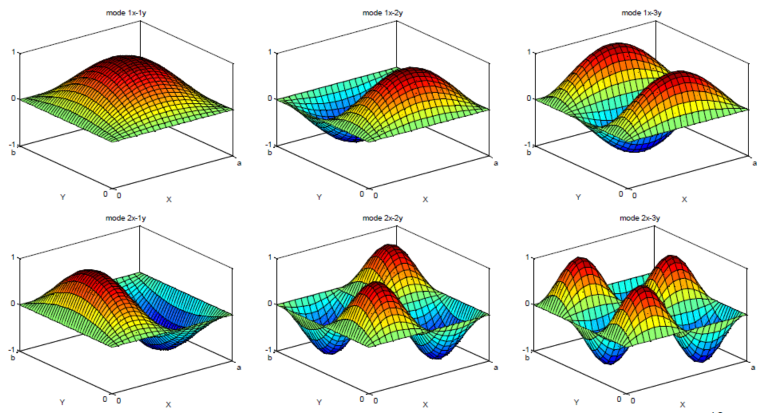


Figure 1.4: First six mode shapes of a thin plate with simply supported edges

For the solution to be non-trivial, the 2×2 coefficient matrix needs to be singular, i.e.

$$(\gamma_1^2 + \gamma_2^2) \sin(\gamma_2 b) \sinh(\gamma_1 b) = 0. \tag{1.34}$$

Since $\gamma_1 \gamma_2 \neq 0$, then $\sin(\gamma_2 b) = 0$ in order for (1.34) to be solved. This implies that $C = 0$, A is arbitrary and $\gamma_2 = \frac{m\pi}{b}$.

Under those conditions, $F(y)$ can finally be defined as

$$F(y) = sin\left(\frac{n\pi}{b}y\right), \quad (n = 1, 2, ...), \tag{1.35}$$

and the spatial term $\Phi(x, y)$ as:

$$\Phi_{n,m}(x, y) = \sin\left(\frac{m\pi}{a}x\right) \sin\left(\frac{n\pi}{b}y\right), \quad (m, n = 1, 2, ...). \tag{1.36}$$

For each combination of m and n, a specific $\Phi_{m,n}(x, y)$ can thus be defined, which represents the mn_{th} plate **mode shape**. In Fig. 1.4 the first six mode shapes of a rectangular plate are shown. Each mode shape will be modulated in time by the harmonic function $G(t)$ defined in (1.20), whose carrier frequency is the mn_{th} **natural frequency** of the system, also calles modal frequency, namely

$$\omega_{m,n} = \left[\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2\right]\sqrt{\frac{B}{\rho s}}, \quad (m, n = 1, 2, ...). \tag{1.37}$$

Recalling the definition of the bending stiffness B in (1.14) and the relation between the temporal and the angular frequency $f = \frac{\omega}{2\pi}$, the expression of the modal frequency $f_{m,n}$ in Hz can be derived as

$$f_{m,n} = 0.453vs\left[\left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2\right], \tag{1.38}$$

where

$$v = \sqrt{\frac{E}{\rho(1 - \nu^2)}}, \tag{1.39}$$

is the waves propagation velocity through the medium.

It is important to recall that (1.37) and (1.38) hold only for rectangular plates with simply-supported edges made of isotropic material. This is actually not the case of wood, the principal material used in violin making, that is an orthotropic material instead. This means that the material properties differ along the three spatial dimensions. Instead of having only one Young's modulus and a Poisson's ratio, an orthotropic material is fully described by three Young's moduli E_i and six Poisson's ratios ν_{ij}, with $i, j \in x, y, z$, related by [29]

$$\frac{\nu_{ij}}{E_i} = \frac{\nu_{ji}}{E_j}. \tag{1.40}$$

The formula for the modal frequencies of an isotropic plate can be thus adapted to the case of orthotropic plates as [29]

$$f_{m,n} = 0.453s\left[v_x\left(\frac{m}{a}\right)^2 + v_y\left(\frac{n}{b}\right)^2\right], \tag{1.41}$$

where $v_x = \sqrt{\frac{E_x}{\rho(1-\nu_{xy}\nu_{yx})}}$ and $v_y = \sqrt{\frac{E_y}{\rho(1-\nu_{xy}\nu_{yx})}}$, are the propagation velocities of waves through the medium along the x and y direction, respectively.

In any case, the plate free response can be obtained as a superposition of the infinite plate natural modes, namely

$$w(x, y, t) = \sum_{m=1}^{M}\sum_{n=1}^{N} \Phi_{m,n}(x, y) \cdot Ge^{j2\pi f_{m,n}t}, \tag{1.42}$$

where $f_{m,n}$ can be defined either as in (1.38) or as in (1.41), depending if an isotropic or orthotropic material is considered, respectively.

This is actually a very central point, because methods such as Experimental Modal Analysis (EMA), that will be deeply discussed in Sec. 1.1.3, use the theory behind modal superposition in order to model the FRFs experimentally measured on continuous systems such as thin plates. This methods acts with the purpose of retrieving the modal parameters of the system under analysis, through proper algorithms that minimize the distance, computed in terms of a specific loss function, between the modeled FRF and the experimental one, in order to understand the properties of the system itself.

1.1.3 Modal Analysis

Also for continuous systems such as thin plates, the concept of FRF introduced in Sec. 1.1.1 for the simpler case of a mechanical oscillator still applies. In this case, we have that the FRF becomes a point-to-point relation that can be defined for each pair of points, i.e. the excitation and the measurement point. In this sense we talk about local FRF and, as in

$$\begin{tikzcd} X(\omega) \arrow[r] & \boxed{H(\omega)} \arrow[r] & Y(\omega) \end{tikzcd}$$

Figure 1.5: Schematic representation of the input/output relation for a system.

our case we analyze the plate response in terms of normal deformation, we talk about **point receptance**.

In the preliminary studies performed during our work we will retrieve displacement data from a plate FEM model to which a normal excitation load is applied, in order to finally obtain the point receptance signals. In real applications performed on musical instruments such as violin plates, this is typically done through the so called hammer test. However, before going in deep in the comprehension of how experimental FRF measurements are taken, it is necessary to give a brief background on the theory of linear systems.

For a linear system, as the one depicted in Fig. 1.5, the output $y(t)$ can be obtained as the convolution of the system input $x(t)$ through the Impulse Response function $h(t)$ proper of the system itself. In our case study, the system is a rectangular thin plate, the input is an impulsive force applied on it, while the output is the plate normal displacement measured on a specific point of the plate surface. In the time domain we thus have that a linear system is fully described by the relation

$$y(t) = h(t) * x(t), \tag{1.43}$$

where $*$ indicates the convolution operation.

If analyzed in the frequency domain, after performing the Fourier Transform of the quantities involved, (1.43) turns into the product

$$Y(j\omega) = H(j\omega) \cdot X(j\omega), \tag{1.44}$$

where $H(j\omega)$ is the Frequency Response Function characterizing the system under study.

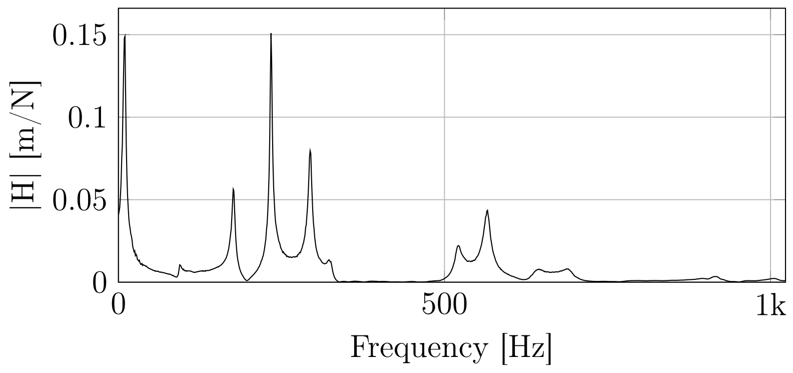
Given that the convolution operation is a time consuming task, what is commonly done when retrieving the FRF of a linear system, is to compute the frequency domain counterparts both of input $X(j\omega)$ and of the output $Y(j\omega)$, obtaining the FRF $H(j\omega)$ by simply inverting (1.44) as

$$H(j\omega) = \frac{Y(j\omega)}{X(j\omega)}. \tag{1.45}$$

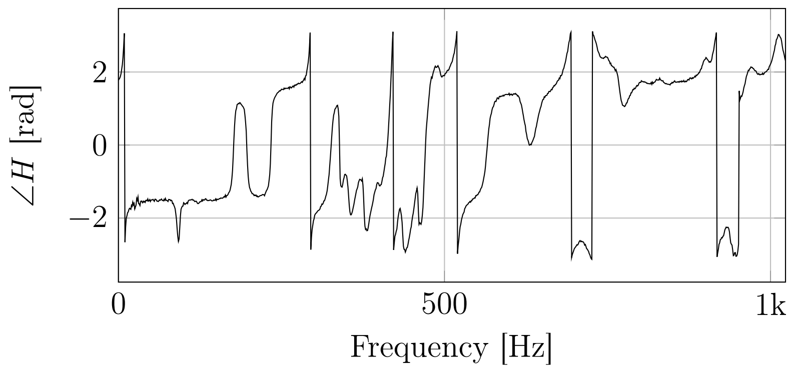
This theory can be fully applied when performing hammer tests for FRF data acquisition on a violin plate. The procedure consists in measuring, with the aid of a laser scanner or a set of accelerometers, the plate response to an impulsive excitation given by an impact hammer in different positions over the plate surface. Local FRFs are then obtained for each measurement point as the ratio between the frequency counterparts of the plate response (local system output) and the excitation force signal (system input) as in (1.45).

As already mentioned in Sec. 1.1.1 indeed, when performing vibration studies on a continuous system as violin plates, a good practice is to treat it as a system with n DOFs, where n is finite. Acquiring FRF data on a plate with a set of n accelerometers actually means discretizing the plate itself as a n DOFs system and for each of the accelerometers positioned over the plate surface, a local FRF can be derived in terms of receptance, mobility, accelerance or impedance, depending on the purpose of the study.

In its most general form, each local FRF is a complex-valued signal and, for this reason, it can be displayed through plots of its magnitude and phase, as shown in Fig. 1.6. In real measurements, FRFs cannot



(a)



(b)

Figure 1.6: Local Receptance acquired from an hammer test conducted on a rectangular plate. In (a) the magnitude signal; and in (b) the phase, both plotted over frequency.

be described simply as the ratio between output and input. The presence of noise or non linear behaviours may lead to some incoherence in the gathered information. To quantify the correlation between the measured output signal and the input, a useful strategy is to evaluate the **coherence function** [30], defined as

$$C_{XY} = \frac{\|S_{XY}\|^2}{S_{XX}S_{YY}}, \tag{1.46}$$

where S_{XX} and S_{YY} are the auto-spectral densities of the input and the output, respectively, and S_{XY} is the cross-spectral density of the input and the output.

Once the experimental FRFs have been obtained, one of the standard approaches to retrieve the system modal parameters is **Experimental Modal Analysis** (EMA) [31], which consists in the approximation of each local FRF as the superposition of the first n modes, where n is the number of accelerometers. Notice that the symbol n we are using here has not to be confused with that one used before to indicate the mode shapes index along the plate x axis. According to the modal superposition approach, the displacement w of a system evaluated in a specific point $\boldsymbol{r_j} = (x_j, y_j)$, due to a monoharmonic load $\tilde{F}_k = F_{k0}e^{j\omega t}$ applied in $\boldsymbol{r_k} = (x_k, y_k)$, can be expressed as a linear combination of its modes of vibration $\Phi_{n,m}$, namely

$$w(\boldsymbol{r_j}, \boldsymbol{r_k}, t) = F_{k0}e^{j\omega t} \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\Phi_{m,n}(\boldsymbol{r_j})\Phi_{m,n}(\boldsymbol{r_k})}{-\omega^2 m_{m,n} + j\omega c_{m,n} + k_{m,n}}, \tag{1.47}$$

where $m_{m,n}$ is the mn_{th} modal mass, $k_{m,n}$ is the mn_{th} modal stiffnesses, and $c_{m,n}$ is the mn_{th} modal damping.

The local FRF $H(\boldsymbol{r_j}, \boldsymbol{r_k}, \omega)$ can thus be found as the sum of contributions of the $M \times N$ vibrational modes. Substituting the expression of w in modal coordinates obtained in (1.47), we can express the FRF as:

$$H(\boldsymbol{r_j}, \boldsymbol{r_k}, \omega) = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\Phi_{m,n}(\boldsymbol{r_j})\Phi_{m,n}(\boldsymbol{r_k})}{-\omega^2 m_{m,n} + j\omega c_{m,n} + k_{m,n}}. \tag{1.48}$$

The dynamic properties of interest are characterized by the modal parameters $m_{m,n}$, $k_{m,n}$, $c_{m,n}$ and $\Phi_{m,n}$. These can be identified by minimizing, usually through a least squares approach, the analytical responses with the measured ones. The modal frequencies $\omega_{m,n}$ associated to the plate can then be identified once the other parameters are found.

Nowadays EMA represents a standard and there is a wide literature available where it is exploited [32][33]. An alternative to it is the **Finite Element Modal Analysis** (FEA), which uses FEM to model the dynamic behaviour of a structure, without passing through preliminary experimental measurement as done by EMA. FEM is particularly useful for solving complicated differential equations with numerical solutions, and works by approximating the shape of a structure with a finite number of smaller geometrical segments, i.e. *elements*, each containing several *nodes*, for which analytical functions can be defined and solved. The overall behaviour of the complete structure is then found by solving these exact functions. Each node in the discretized geometry will have several DOFs. For instance, a 4-node-8-DOF element can have either two translational DOFs or one translational DOF and one rotational DOF on each node. To interpolate the computed values between the nodes, a shape function is used, which usually has the form of a polynomial.

To understand the principles of FEM, let's consider a generic differential equation, namely

$$\zeta(u(x), u(\dot{x}), u(\ddot{x})) = 0. \tag{1.49}$$

What FEM does actually is to approximate the solution $u(x)$ of the problem by expressing it as the corresponding Taylor expansion truncated to the n^{th} order, namely

$$u(x) \approx a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n = \sum_{i=1}^{n} a_i x^i. \tag{1.50}$$

This approximate solution can also be represented with a set of DOFs at the nodes

$$u(x) \approx N_1(x)u_1(x) + N_2(x)u_2(x) + ... + N_n(x)u_n(x) = \sum_{i=1}^{n} N_i u_i(x), \tag{1.51}$$

where N_i is the shape function at node i and $u_i(x)$ is the DOF at node i. A feature of the shape function is $N_i(x) = 1$ at node i, and $N_i(x) = 0$ at other nodes. In this manner, each shape function can represent the weight of the corresponding node i.

When modeling 2D and 3D structures, 2D and 3D elements are used, but in any case, the main point is always to find an expression for the shape functions that will be used to calculate the mass and stiffness matrices of the element. As already mentioned before, when performing vibration studies on a complex structure, by assuming the problem as linear, the structure itself can be considered as the sum of a bunch of harmonic oscillators, one for each DOF. Then the equation of motion (1.6) can be discretized and written in matrix form as

$$[M][\ddot{X}] + [C][\dot{X}] + [K][X] = [F], \tag{1.52}$$

where M is the mass matrix, C is the damping matrix, K is the stiffness matrix and F is the external force vector.

If the vibrating structure has a complex shape and intricate boundary conditions, it is hard to accurately define the mass, damping and stiffness matrices, as well as the loads and the boundary conditions in (1.52). It will be more straightforward to represent the complex geometry with a set of elements, each of those will have its own mass, stiffness, and damping properties. By assembling the small matrices associated to each element, expressed in terms of the shape functions N_i, into the big matrices in (1.52), all the natural frequencies and displacements at each node can be calculated.

In [34], FEM and EMA are compared when applied to vibration studies on a violin top plate made of spruce. In this work, a FEM model emulating a real spruce top plate was first built and FEA was applied in order to calculate the natural frequencies and mode shapes. Then

EMA was also performed on the real plate, to get not only the natural frequencies and mode shapes, but also the damping ratios of each mode. The two sets of results were carefully compared, proving that they fitted well with each other. This actually proves that FEM models, as that one proposed in [34], are reasonably accurate, and this, in the specific field of violin making, can be helpful also for luthiers who may want to modify their designs to achieve an even better sound quality.

There are many commercial software exploiting FEM modeling. One of those is COMSOL Multiphysics®, that is indeed the one we are going to use to generate our dataset of synthetic FRFs starting from the model of a thin plate, as will be deeply discussed in Chapter 3.

1.2 State of the Art

A typical issue that vibration studies have to face with, is the choice of the right spatial sampling period when acquiring vibration data to retrieve FRF measurements from a system, i.e. the spatial step that separates two adjacent measurement points. As formalized by the Nyquist-Shannon theorem [35], in order to gain information on a sufficiently wide frequency band, it is necessary to sample the space with a spatial frequency f_s that cannot be less than the double of the highest detectable frequency f_{MAX}, namely

$$f_s \geq 2 f_{MAX}. \tag{1.53}$$

The problem is that, during real acquisitions, it is hard to respect the constraint in (1.53), most of the time due to the high costs related to data acquisition or to the impossibility of placing sensors in some specific areas of the system surface. This leads to measurements affected by poor spatial resolution. As will be better discussed in Sec. 2.1, vibration measurements acquired from a rectangular thin plate approximable as a 2D structure, can be represented as 3D tensors $\mathbf{H}$, where FRF data are collected as a function of both space and frequency, and usually those tensors are sliced to visualize FRFs in terms of 2D images. A typical visualization consists in plotting the FRF data for all the spatial measurement points in correspondence of a specific frequency value, in order to analyze the system shape for a specific excitation frequency. Sampling issues encountered during experimental measurements result therefore in images with poor resolution.

Is that for this reason that super-resolution (SR) algorithms could help in this sense, to retrieve information and reconstruct missing data on those images with a poor definition, being able to overcome the high costs related to data acquisition. SR is a popular technique for increasing the resolution of a given image. Its most common application is to provide better visual effects after resizing a digital image for display or printing. In recent years, imaging and display devices have become ubiquitous, and image SR is thus of primary importance. SR is considered part of the so-called inverse problems, i.e. problems involving the estimation of parameters or data from inadequate observations often noisy and containing incomplete information about the target data, usually due to physical limitations in the measurement devices.

SR methods are traditionally categorized according to the number of input images used: when several low-resolution (LR) input images are available we talk about Multi Frame Super-Resolution (MFSR); on the other hand, when the LR input image is unique, we talk about Single Image Super-Resolution (SISR). In both cases a unique High-Resolution (HR) image is produced as output.

MFSR methods, which are well presented in [36], have been studied since the SR problem first appeared in the scientific community [37]. Here, the multiple LR input images are assumed to be different views of the same scene, taken with sub-pixel misalignments, i.e. each image is seen as a degraded version of an underlying HR image to be estimated, where the degradation processes can include blurring, geometrical transformations, and downsampling. SISR, instead, aims at constructing the HR output image from a single LR input image and is deeply connected with traditional analytical interpolation, since they share the same goal.

Traditional interpolation methods implicitly impose a smoothness prior by computing the missing pixels in the HR grid as weighted combinations of known pixels, either linear or non linear depending on the specific method considered. However, natural images often present strong discontinuities, such as edges and corners, and thus the smoothness prior results in producing ringing and blurring artifacts in the output image. SISR algorithms can be broadly classified into two main approaches: *interpolation-based methods* , which follow the interpolation approach by posing more sophisticated statistical priors, possibly in a non-parametric fashion; *data-driven methods*, which instead perform interpolation by learning patterns directly from data.

1.2.1 Interpolation-based algorithms

Interpolation-based methods, also known as model-based methods, generate a HR image from its LR version by estimating the pixel intensities through a suitable interpolation scheme. Taken a LR image $I^{(\mathrm{LR})}_{i,j} \in \mathbb{R}^{W\times H}$, its corresponding HR version is $I^{(\mathrm{HR})}_{i,j} \in \mathbb{R}^{aW\times bH}$, where a and b are magnification factors of the width and the height respectively. Without loss of generality, we assume $a = b = 2$. We can easily get pixel values of $I^{(\mathrm{HR})}_{2i,2j}$ from the low-resolution, imposing

$$I^{(\mathrm{HR})}_{2i,2j} = I^{(\mathrm{LR})}_{i,j} \ (i = 0,1,\dots,H;\ j = 0,1,\dots,W). \tag{1.54}$$

Interpolation is then to get the set of pixel values denoted as $I^{(\mathrm{HR})}_{2i+1,2j}$, $I^{(\mathrm{HR})}_{2i,2j+1}$ and $I^{(\mathrm{HR})}_{2i+1,2j+1}$. As shown in Fig. 1.7, black nodes denote the pixels which can be directly obtained from the LR image, while white

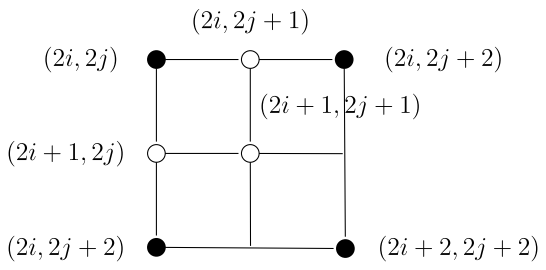


Figure 1.7: Pixels in the high-resolution image. Black nodes denote pixels directly obtained from the low-resolution image. White nodes denote interpolated pixels

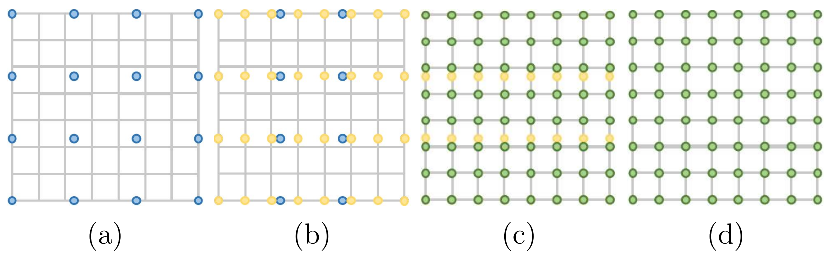


Figure 1.8: Interpolation-based upsampling methods. The gray board denotes the coordinates of pixels, and the blue, yellow and green points represent the initial, intermediate and output pixels, respectively.

nodes denote the pixels which are unknown and can be gained through interpolation schemes.

The most common interpolation methods used in practice are *nearest-neighbour* (NNI), *bilinear* (BLI) and *bicubic* (BCI) [38, 39]. NNI is a simple and intuitive algorithm. It selects the value of the nearest pixel for each position to be interpolated regardless of any other pixel. Therefore, this method is very fast but usually produces discontinuous results with low perceptual quality. BLI first performs linear interpolation on one axis of the image and then along the other one, as Fig. 1.8 shows. Since it results in a quadratic interpolation with a receptive field sized 2×2, it shows much better performances than NNI while keeping relatively low computational costs. Similarly, BCI [40] performs cubic interpolation on both the two axes, as depicted in Fig. 1.8. Compared to BLI, BCI takes 4×4 pixels into account, and results in smoother images with fewer artifacts, with the drawback of comporting higher computational costs. BCI produces noticeably sharper HR images than BLI and NNI, and is perhaps the ideal combination of processing time and output quality. For this reason it is a standard in many image editing programs (including Adobe Photoshop), printer drivers and in-camera interpolation. However, all these three interpolation methods, since they are based on an oversimplified slow varying image model, often produce images with various problems along the boundaries of objects depicted, including aliasing, blurring, and zigzagging edges.

To overcome these problems, various algorithms have been proposed

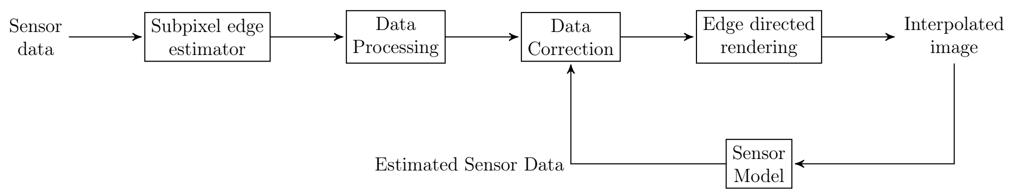


Figure 1.9: Architecture of the edge directed interpolation method

to improve the interpolation-based approaches and reduce edge artifacts, aiming at obtaining images higher definition along the edges. In [41] a method has been proposed for estimating the orientation of each single edge in the image by using projections onto an orthonormal basis and the interpolation process has been modified to avoid interpolating across the edge. Each edge is modelled by using a function of four parameters, namely

$$S(i,j,A,B,\rho,\theta) = \begin{cases} A \;\; if\; i\cos\theta + j\sin\theta \geq \rho \\ B \;\; if\; i\cos\theta + j\sin\theta < \rho \end{cases}, \tag{1.55}$$

where i an j represent the coordinates on a Cartesian plane, ρ and θ are polar coordinates, and $i\cos\theta + j\sin\theta = \rho$ specifies a straight line separating the two regions of constant intensity value A and B respectively. An orthogonal basis $B_n(\phi)$ is defined then, and the projection of the edge model S onto $B_n(\phi)$ yields to a set of spectral coefficients a_n from which the edge orientation θ can be computed. This approach has shown to perform well in correspondence of homogeneous regions within the original images; however, the reconstructed images resulted to be blurred in the presence of fine edges.

Allebach et al. proposed in [42] to generate a HR edge map starting from the LR image, using it to guide the interpolation. Two steps characterize this method: rendering and data correction. Rendering is a modified version of BLI applied on the LR data. Once the interpolated image is obtained, it is fed back through the sensor model, adjusting the mesh values on which the BLI is based, by measuring the disparity between the resulting estimated sensor data and the true sensor data. The edge directed interpolation yields a much sharper result than the method of Jelsen et al. [41], while some aliasing artifacts, even if not so prominent, are still present. Fig. 1.9 shows the architecture of the edge directed interpolation technique itself.

Other proposed methods perform interpolation in a transform (e.g. wavelet) domain [43, 44]. These algorithms assume the LR image to be the low-pass output of the *wavelet transform* and exploit dependencies across wavelet scales to predict the missing coefficients in the higher range of the domain. In [44], Carey et al. made use of the wavelet transform as a mean through which quantify the signal local smoothness; the mathematical smoothness (or regularity) is bounded by the rate of decay of its wavelet transform coefficients across scales. The algorithm proposed creates new wavelet subbands by extrapolating the local coefficient decay. These new, fine scale subbands are used together with the original wavelet subbands to synthesize an image of twice the original size. The extrapolation of the coefficient decay preserves the local regularity of the original image, thus avoiding over smoothing problems.

As a matter of fact, interpolation-based SR methods improve the image resolution only basing the operation on the image signals, without bringing any more information. It has also to be said that they often introduce some side effects, such as high computational costs, noise amplification and blurred outputs. Therefore, the current trend is to replace interpolation-based methods with data-driven ones.

1.2.2 Data-driven algorithms

Data-driven SR methods, also known as learning-based methods, have been brought into focus because of their outstanding performances. They are usually based on the use of Machine Learning (ML) to analyze statistical relationships between the LR image and its HR counterpart from substantial training examples. Data-driven methods consist either in pixel-based procedures where each value in the HR output image is singularly inferred via statistical learning [45, 46], or patch-based procedures where HR estimation is performed thanks to a dictionary of correspondences between LR and HR patches, i.e. squared blocks of image pixels. The data-driven methods making use of patches are also referred to as *example-based* SR methods [47]. During the upscaling procedure, the LR input image is thus divided into patches, and for each LR input patch a single HR output patch is reconstructed, by observing the examples contained in the dictionary.

As the history of data-driven methods is concerned, the *Markov random field* (MRF) approach was first adopted by Freeman *et al.* in [47], to exploit the abundant amount of real-world images to synthesize visually pleasing image textures. *Neighbor embedding methods*, proposed by Chang *et al.* [48], took advantage of similar local geometries between LR and HR data in order to restore HR image patches. Lately, sparse signal recovery theory [49] have been applied to SR problems. Sparse representations encode a signal vector $\boldsymbol{x}$ as a linear combination of few atoms included a dictionary $\boldsymbol{D}$, i.e.

$$\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{\alpha}, \tag{1.56}$$

where $\boldsymbol{\alpha}$ is the sparse coding vector. As for SISR, Yang et al. first proposed a Sparse coding Super Resolution (ScSR) method in [50]. During the training phase, given a group of LR and HR training patch pairs, ScSR aims to jointly learn an HR dictionary $\boldsymbol{D^h}$ and an LR dictionary $\boldsymbol{D^l}$ to reconstruct the HR and LR patches by assuming that each LR/HR patch pair shares the same sparse coding vector. In the testing phase, the image is divided into overlapped patches, and each patch is encoded by means of the LR dictionary $\boldsymbol{D^l}$ with sparse coefficient $\boldsymbol{\alpha}$. The corresponding HR patch is reconstructed by $\boldsymbol{D^h}$ and $\boldsymbol{\alpha}$ with $\boldsymbol{D^h\alpha}$. Finally, the HR image can be obtained by aggregating all the estimated HR patches into a whole image. Inspired by ScSR [50], many sparse coding and dictionary learning based methods have been proposed for SISR. By relaxing the constraint that the LR/HR patch pair has the same coding vector, Wang et al. [51] introduced a transform matrix to allow more complex relationships between the HR and LR coding vectors, and proposed a semi-coupled dictionary learning (SCDL) method for SISR. Subsequently, more complex models have been proposed for better modeling the relationship between the LR and HR spaces with coupled dictionaries. He et al. [52] presented a non-parametric Bayesian approach to learn dictionaries to build relationship between the LR and HR spaces. Peleg and Elad [53] proposed a statistical model which uses restricted Boltzmann machine (RBM) to model the relationship between the LR and HR coding vectors. Zhu et al. [54] suggested to enhance the flexibility of the HR dictionary by permitting certain deformation in each HR patch.

Lately, random forest [55] has also been used to improve the reconstruction performances. Meanwhile, many works combined the merits of reconstruction-based methods with the learning-based approaches to further reduce artifacts introduced by external training examples [56, 57, 58, 59]. In [57], a unified SR framework is presented containing two stages, learning and reconstruction, to seamlessly take advantages of both learning-based and reconstruction-based methods. The work is based on the observation that local patches in natural images tend to repeat themselves many times both within the same scale and across different scales. The redundancy across different scales requires that local patches in the desired HR image should be similar to those in the input LR image. With this requirement, [57] assumes that the local patches in the desired HR image and those in the input LR image admit a sparse approximation over the same dictionary learned from the input LR. A regularization term is built by employing the Normalized Least Mean Squares (NLMS) adaptive filter to sharpen edges and suppress artifacts. A single dictionary is learnt from the input LR image itself and an hallucination regularization term and a non-local regularization term are constructed over the prediction to make the SR problem well-posed. At first, sparse representation over a self-learned dictionary is employed to form a learning-based regularization term (the hallucination regularization term) and then a NLMS is introduced, to develop another reconstruction-based regularization term (the non-local regularization term). By incorporating the above two regularization terms into a Maximum A Posteriori (MAP) based energy function, a local optimal solution is obtained by using the gradient descent algorithm.

Nowadays, deep learning models and their applications in the field of image processing are of central interest even in the specific context of SR problems, as they have demonstrated great superiority to reconstruction-based and other data-driven methods. Particularly interesting results have been achieved by means of Convolutional Neural Networks (CNNs) [60, 61, 62], which are for sure the main topic of interest in these years in many fields of study, due to their strong capability to learn valid features from big data in an end-to-end manner. The purpose of this work is indeed to follow this trend, going beyond in the application of CNNs to SR problems, trying to see their efficiency in the specific case study of FRF data retrieval.

1.3 Neural Networks

The aim of this section is to give the reader an overview on the ML techniques used in this work. There will be a first introduction to ML, a branch of computer science that provides systems the ability to learn valid, novel, potentially useful and understandable features directly from data. We will focus then on a specific branch of ML, i.e. Deep Learning, introducing the reader to Artificial Neural Networks (ANNs). We will start from the basic component, i.e. the perceptron, moving to the definition and the analysis of Multi-layer perceptrons, finally going deeply into the classification of the most common ANN architectures: Feed-Forward Neural Networks, convolution Neural Networks (CNNs) and Convolutional Autoencoders (CAs). An overview of the main regularization techniques when training ANNs will also be given and finally we will introduce the architecture proposed in this work, i.e. the U-net, a particular CA designed for image segmentation.

1.3.1 Background on Machine Learning

With Machine Learning we intend that category of algorithms focused on finding patterns and learning features directly from data, using the knowledge achieved to make valuable predictions that will depend on the specific task considered. ML falls within the Artificial Intelligence (AI) umbrella, which in turn intersects the broader field of knowledge discovery and data mining. The basic idea is to build a statistical model from input data and use it to make predictions, or decisions, on new observations. Given a certain experience E, i.e. a dataset $\mathcal{D} = \{x_1, x_2, x_3, ..., x_N\}$, there are three main ML paradigms:

- **Supervised Learning**: given the desired outputs $\{t_1, t_2, t_3, ..., t_N\}$, the algorithm learns to produce the correct outputs related to a new set of inputs;

- **Unsupervised Learning**: the algorithm exploits regularities in $\mathcal{D}$, building a representation to be used for reasoning or prediction;

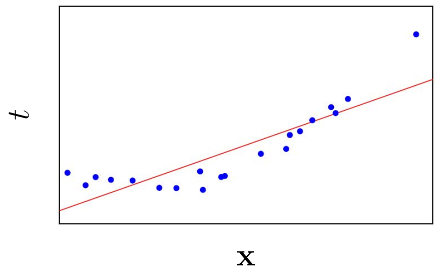


Figure 1.10: Linear Regression model

- **Reinforcement Learning**: the algorithm learns to produce actions $a_1, a_2, a_3, ..., a_N$ affecting the environment, in order to maximize the rewards $r_1, r_2, r_3, ..., r_N$ in the long term.

The problems tackled by ML are basically of two types:

- **Regression Problems**: The goal of regression is to learn a mapping from input $\mathbf{x}$ to a continuous target output. The simplest case is represented by *linear regression*, where the model is a linear input-output relationship, namely

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{D-1} w_j x_j = \mathbf{w}^T \mathbf{x}, \tag{1.57}$$

where $\mathbf{x} = (1, x_1, ..., x_{D-1})$ is the *input vector* and $\mathbf{w}$ is the *parameters vector*. In Fig. 1.10 a linear regression model is shown.

The model above can be improved introducing nonlinear *basis functions* inside the combination, namely:

$$y(\Phi, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}), \tag{1.58}$$

where $\Phi(\mathbf{x}) = (1, \phi_1(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x}))^T$.

Examples of basis functions could be Polynomials $\phi_j(x) = x^j$, Gaussians $\phi_j(x) = exp(-\frac{(x-\mu_j)^2}{2\sigma^2})$ and Sigmoidals $\phi_j(x) = \frac{1}{1+exp(\frac{\mu_j - x}{\sigma})}$. The space of the basis functions is called *feature space*. In this case, we still talk about linear regression, as the relation between the feature vector $\Phi(\mathbf{x})$ and the prediction $y(\Phi, \mathbf{w})$ is clearly linear. The fundamental difference is that now linearity is no more in the input space but in the feature space. This is useful when it is necessary to deal with input and output nonlinear relations, as shown in Fig. 1.11. The passage from input space to feature space indeed is performed in order to convert a nonlinear complex model used to fit data in the original input space into a simpler linear one in the high-dimensional feature space. This transformation spreads the data out in such a way that a linear hyper-plane can be fitted.

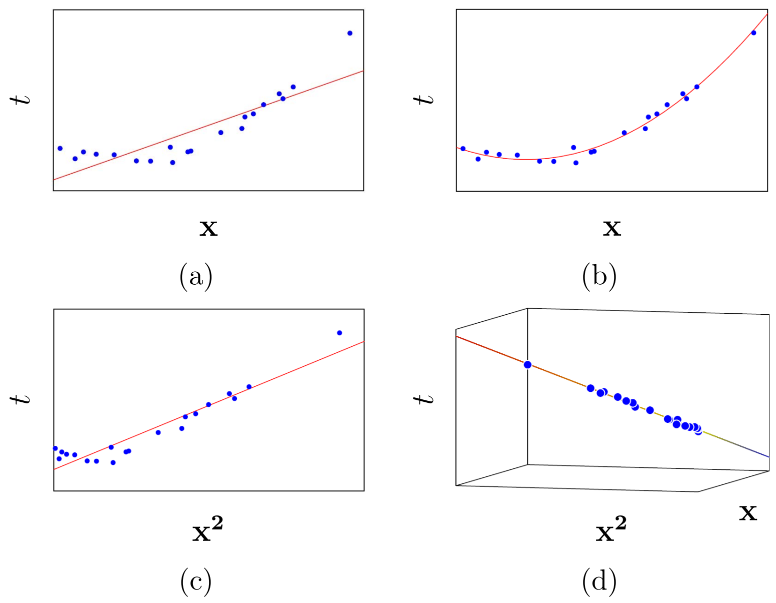


Figure 1.11: Polynomial basis functions $\phi_1(x) = x$ and $\phi_2(x) = x^2$ used as an augmentation of the original input space.

- **Classification Problems**: The goal of classification is to assign an input $\mathbf{x}$ to a specific class C_k, where $k = 1, ..., K$. Typically, each input is assigned only to one class and the input space is divided into decision regions whose boundaries are called *decision boundaries* or *decision surfaces*. While a linear regression model is used to predict a continuous output and is linear with respect to the parameters vector $\mathbf{w}$, in classification problems we need to predict discrete class labels, or posterior probabilities that lie in the range $(0, 1)$, so we need a nonlinear function, also called *activation function*, changing (1.57) into

$$y(\mathbf{x}, \mathbf{w}) = f\left(w_0 + \sum_{j=1}^{D-1} w_j x_j\right) = f(\mathbf{w}^T \mathbf{x}). \tag{1.59}$$

As for regression problems, it is possible to make a fixed nonlinear transformation of the input space using a vector of basis functions $\Phi(\mathbf{x})$, passing to a feature space. (1.59) thus become

$$y(\Phi, \mathbf{w}) = f\left(w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})\right) = f(\mathbf{w}^T \Phi(\mathbf{x})). \tag{1.60}$$

In classification problems, the result of this operation is that complex nonlinear decision boundaries in the original input space can be translated into linear boundaries in the feature space, as can be seen in Fig. 1.12. The simplest case of classification problems is *binary classification*, where the nonlinear function used is a *binary*

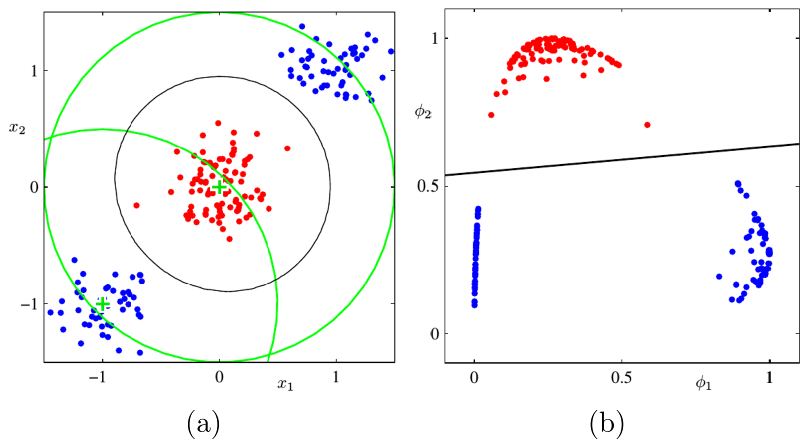


Figure 1.12: Graphical representation of a binary classification problem. Data are plotted (a) in the original input space, and (b) in the new feature space, after having applied two Gaussian basis functions ϕ_1, ϕ_2

step function defined as

$$y(\Phi, \mathbf{w}) = \begin{cases} \textit{1 if } \mathbf{w}^T\Phi(\mathbf{x}) > 0 \\ \textit{0 otherwise} \end{cases} . \tag{1.61}$$

As can be imagined, this function will not be useful in multi-class classification problems and this is actually a non trivial limitation. Moreover, the gradient of the binary step function equals zero for each input $\mathbf{x}$, which causes a hindrance in processes like gradient descent that will be better discussed lately. For classification problems it is thus preferred to use the so called *logistic function* (Sigmoid), as it is nonlinear and continuously differentiable. It is defined as

$$y(\Phi, \mathbf{w}) = \frac{1}{1 + \exp\left(-\mathbf{w} \cdot \Phi(\mathbf{x})\right)}. \tag{1.62}$$

In Fig. 1.13 both a step function and a sigmoid are shown.

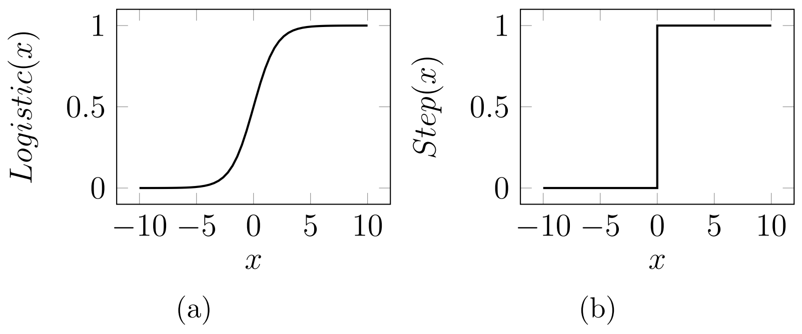


Figure 1.13: Graphical comparison of two well known activation functions. In (a) a Sigmoid and in (b) a binary step function

The purpose of a ML model is to learn a function $\mathcal{F}$, such that

$$\hat{y}_n = \mathcal{F}(\mathbf{x}_n, \mathbf{w}) \approx y_n, \tag{1.63}$$

where $\mathbf{x}_n$ is the n_{th} input vector in the dataset $\mathcal{D}$, y_n the corresponding target output, $\hat{y}_n$ is the model prediction and finally $\mathbf{w}$ is the matrix of the model weights. Weights $\mathbf{w}$ are learned from data (hence the name data-driven methods) such that a measure of the distance between the prediction $\hat{y}_n$ and the desired output y_n is minimized. The learning process is thus based on the definition of a proper **loss function** $L(y_n,\hat{y}_n)$. The choice of $L(\cdot)$ is one of the most crucial points in ML algorithm design, given that the performances of the model will depend on its definition. The type of loss function to be chosen is strictly related both to the type of problem to be addressed and in particular to the data space. In the literature a great variety of loss functions is present, but still, as we will do in this work, it is possible to define new losses also by making a combination of the most used in the literature.

For what concerns regression problems, the most used loss function is the Mean Squared Error (MSE), namely

$$\text{MSE} = \frac{1}{N}\sum_{n=0}^{N-1}(y_n - \hat{y}_n)^2, \tag{1.64}$$

where N indicates the number of samples in the dataset $\mathcal{D}$. MSE is thus a measure of the mean value of the square differences between the predictions $\hat{y}_n$ and the target outputs y_n. Usually MSE is normalized by the mean square value of the target output. In this case we talk about Normalized Mean Squared Error (NMSE)

$$\text{NMSE} = \frac{\sum\limits_{n=0}^{N-1}(y_n - \hat{y}_n)^2}{\sum\limits_{n=0}^{N-1} y_n^2}. \tag{1.65}$$

For classification problems instead, the most commonly used loss function is the Categorical Cross-Entropy (CCE), defined as

$$\text{CCE} = -\sum_{n=0}^{N-1}\sum_{c=1}^{C} y_{cn}\log \hat{y}_{cn}, \tag{1.66}$$

where C is the number of classes. With $\hat{y}_{cn}$ we refer to the predicted probability of the n_{th} sample to belong to class c, while y_{cn} is the actual value of the n_{th} sample that is equal to 1 only if it belongs to the class c. In this sense, CCE represents the difference between the probability distribution of the predictions and the targets.

ML models are optimized by means of iterative optimization methods, and the most used one is **Gradient Descent algorithm** [63][64][65].

Gradient Descent is used to find the weight vector $\mathbf{w}$ that minimize the loss function. The procedure starts by defining an initial weights vector $\mathbf{w_0}$. This could contain all zeros or small random values. A cost function E is evaluated by plugging the actual value of the coefficients into the function itself, leading to

$$E_0 = E(\mathbf{w}_0). \tag{1.67}$$

The gradient of the cost function is then computed in correspondence of $\mathbf{w}_0$, as

$$\delta_0 = \left.\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}_0}. \tag{1.68}$$

The gradient is a concept from calculus and refers to the slope of a function at a given point. We need to know the direction where the slope of the cost function is maximum, i.e. where its gradient is maximum. The weights values have to be modified to follow that specific direction, in order to get a lower cost on the next iteration. Now that we know from the gradient which direction is downhill, the weights values can be updated. A learning rate parameter α must be specified to control how much the weights can change during each update. It can be chosen as constant or iteration-dependent.

This process is iterated until the cost function reaches its minimum. The general formula describing the k_{th} iteration is

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \left.\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}_k}. \tag{1.69}$$

If the dataset used for training is entirely used to compute the gradient of the cost function, we talk about Batch Gradient Descent [66]. However, it is also possible to choose a smaller batch size, that is the number of training samples which are randomly drawn, at each iteration, to estimate the gradient. In this case, we refer to the algorithm as Mini-batch Gradient Descent. In the case limit where the batch size is set to only one sample, we talk about **Stochastic Gradient Descent** [66]. Actually batch size is an hyper parameter on which could depend the performances of a ML algorithm. On one extreme, using a batch equal to the entire dataset guarantees convergence to the global optima of the model. However, this is at the cost of slower, empirical convergence to that optima. On the other hand, using smaller batch sizes have been empirically shown to have faster convergence to good solutions. This is intuitively explained by the fact that smaller batch sizes allow the model to start learning before having to see all the data. The drawback of using a smaller batch size however, is that the model is not guaranteed to converge to the global optima. Therefore, under no computational constraints, it is often advised that one starts at a small batch size, reaping the benefits of faster training dynamics, and steadily grows the batch size through training, also reaping the benefits of guaranteed convergence.

In all ML applications, what happens is that the entire dataset is not used just as it is, but it is usually split into three main subsets, which are:

- **Training set**: the set of samples used during the optimization, i.e. to find the model parameters that give the best-fit for the data;
- **Validation set**: samples that are used in parallel while the model is training, in order to see if the model itself is overfitting or not. More precisely, the model weights are iteratively applied on the validation samples, to quantify, through a properly defined loss function, the error between the predictions and the corresponding targets and to evaluate the performances on data unseen during training while training is still going on;
- **Test set**: The samples of the dataset used to provide an unbiased evaluation of the final model on new data never seen before. This set is fundamental because gives an estimate of the model generalization capabilities.

During the training process, we can only act by minimizing the average value of the loss function on the training set, known as training error, defined as

$$E_{train} = \frac{1}{N}\sum_{n=0}^{N-1} E_n, \tag{1.70}$$

where N is the number of samples in the training set and E_n is the error related to the n_{th} sample. However, we would like to minimize also the test error E_{test}, whose value is unknown and whose behaviour is not corresponding to the one of E_{train}. For this reason, we introduce the validation set and we use the average error on this latter, E_{val}, to get an estimate of the test error and refine the performances of our ML model.

The model built during the training phase can be too simple to describe the problem addressed and this situation is known as **underfitting**. Possible solutions in order to avoid that a ML model underfits could be either to increase the size or the number of the weight (parameters) involved, or to increase the complexity of the model itself, or even to extend the training time until cost function is minimized. On the other hand, if the model is too complex, it will learn not only the relevant features, but also the noise contained in the training samples. It will thus be unable to generalize and perform well on new samples. This issue is known as **overfitting** and can be a relevant problem especially on small datasets. There are several ways to avoid overfitting, such as reducing the complexity of the model, increasing the size of the dataset or use some regularization techniques, as those described in Sec. 1.3.5. The number of batches in which the dataset is split gives the number of iterations necessary to complete one **epoch** of training. An epoch is one complete training cycle through the entire training set.

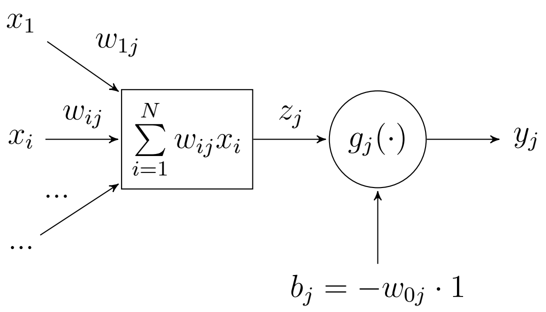


Figure 1.14: Architecture of a perceptron

1.3.2 Basics of deep learning

Deep Learning (DL) and Artificial Neural Networks (ANNs) are currently driving some of the most ingenious inventions in today's century. Their incredible ability to learn from data makes them the first choice for ML scientists when huge datasets are available and complex models are needed in order to fit those data. The power of ANNs consists in their ability to operate with a kind of blind behaviour, producing meaningful results independently of the specific type of data that needs to be managed and learned, and this is actually what makes them so versatile. The name Artificial Neural Network comes from the fact that these ML algorithms are actually an artificial emulation of human beings' nervous system, which can be seen as a network of basic elements called neurons. Each neuron has some dendrites taking inputs from other neurons in the form of electrical impulses, a cell body which generates inferences from those inputs and decide what action to take, and finally an axon terminal which is responsible for the output transmission again in the form of electrical impulses.

ANNs are strictly based on the concept of artificial neuron, also known as **perceptron**. Just like in the human nervous system, a perceptron follows the feed-forward model, meaning that inputs are sent into the cell, are processed, and result in an output to be sent to other perceptrons it is connected with . As it is shown in Fig. 1.14, each perceptron receives some inputs x_i which are then linearly combined through the so called **synaptic weights** w_{ij}, thus obtaining $z_j = \sum_{i}^{N} w_{ij}x_i$. Usually a **bias term** $b_j = w_{j0} \cdot 1$ is added to the linear combination of the inputs and finally a nonlinear function $g(\cdot)$, also called **activation function**, is applied. The final output of the perceptron is thus

$$y_j = g_j\left(\sum_{i=1}^{N} w_{ij}x_i + b_j\right) = g_j\left(\sum_{i=0}^{N} w_{ij}x_i\right). \tag{1.71}$$

Nowadays, the mostly used activation function for ANNs is the Rectified Linear Unit (ReLU), defined as

$$\text{ReLU}(x) = max(0, x). \tag{1.72}$$

The reason why ReLU is preferred to nonlinear functions as the Sigmoid, already introduced in Sec. 1.3.1, is because of its nearly linear behaviour. Sigmoid indeed, as nonlinear functions in general, has the benefit of allowing the perceptrons to learn more complex structures in the data. However, the main problem concerning Sigmoid is that it saturates. This means that large values snap to 1 and small values snap to 0. This function is only really sensitive to input values changing around its mid-point, i.e 0.5. Limited sensitivity and saturation happen regardless of whether the summed activation from the perceptron provided as input contains useful information or not. Once saturated, it becomes challenging for the learning algorithm to continue to adapt the weights to improve the performances of the model. On the other hand ReLU overcomes the saturation issue and, thanks to its nearly linear behaviour, it preserves many of the properties that make linear models easy to optimize with gradient-based methods. It also preserves many of the properties that make linear models generalize well. A graphical representation of ReLU is given in Fig. 1.15.

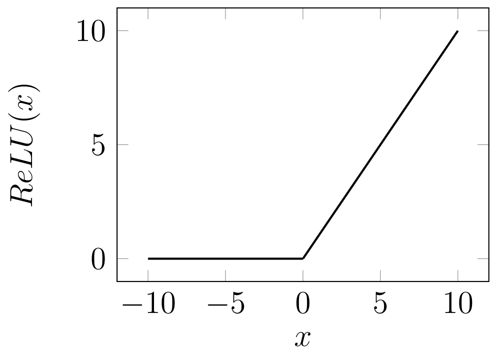


Figure 1.15: ReLU function

The connection of several perceptrons, disposed on a finite number of layers according to a specific topology, gives as a result the so called a Multi-layer Perceptron, that is more commonly named **Feed-Forward Neural Network**, represented in its more general form in Fig. 1.16. Feed-forward Neural Networks consist in an **Input Layer**, composed by

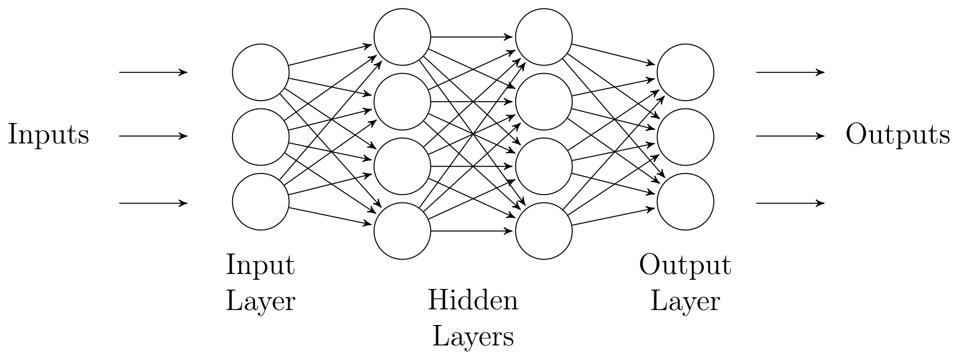


Figure 1.16: Feed-forward Neural Network

those neurons that receives as input the data to process, an **Output Layer** that gives the final results, and a finite set of **Hidden Layers**, which are located between the input and the output layers. The role of each hidden layer is to process data coming from the layer before through a nonlinear combination and give as a result data to be processed by neurons of the next layer. Here for simplicity we refer to Feed-forward Neural Networks with the more generic name ANN. Considering an ANN

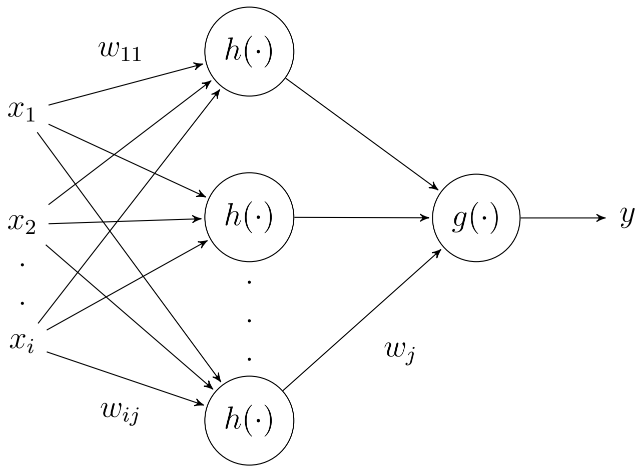


Figure 1.17: Single-hidden layer ANN

with a single hidden layer and only one output, as the one shown in Fig. 1.17, the output y of the network itself will be the linear combination of the outputs of the hidden layer perceptrons, to which an activation function is applied, namely

$$y = g\left(\sum_{j}^{J} w_j \cdot h\left(\sum_{i}^{I} w_{ji} \cdot x_i\right)\right), \tag{1.73}$$

where J is the number of perceptrons in the hidden layer, I is the number of inputs, $g(\cdot)$ is the output activation function and $h(\cdot)$ is the activation function defined for the hidden units.

As done for any other ML model, also for ANN the process of learning is based on a tuning procedure of the network parameters (weights) w_{ij}. We have already introduced in Sec. 1.3.1 the concept of Gradient Descent algorithm, that, in the specific case of an ANN, results in the minimization of the loss function with respect to the computed weights and the biases of all the neurons. However, the computation of the gradient for hidden perceptrons is not a straightforward task, as their outputs are actually unknown.

Back propagation [67] is an efficient algorithm able to solve the issue, as it gives a recursive rule to compute the gradient of the loss function with respect to all the network parameters. The name comes from the fact that the first gradient to be calculated is the one at the output layer and then the computation proceeds backwards through the network layers. Besides, partial computations of the gradient at one layer are reused in the computation of the gradient at the previous one. For the sake of simplicity, we explain the intuition behind the algorithm for Feed-forward Neural Networks, but generalizations exist for other types of architectures.

Let us consider a perceptron u in the analyzed network and let us denote as $pred(u) = (p_1, ..., p_n)$ the set of its predecessors, i.e. the perceptrons in the previous adjacent layer. The weights vector $\mathbf{w}_u$ can be defined as $\mathbf{w}_u = (-b_u, w_{up_1}, ..., w_{up_n})$, where $\mathbf{w}_{up_i}$ denotes the weight of the connection between u and its predecessor p_i and b_u is the bias term. The gradient of the error function with respect to the weights of neuron u can be expressed as

$$\frac{\partial E}{\partial \mathbf{w}_u} = \left(-\frac{\partial E}{\partial b_u}, \frac{\partial E}{\partial w_{up_1}}, ..., \frac{\partial E}{\partial w_{up_n}} \right) = -2\delta_u i_u, \tag{1.74}$$

where $i_u = (1, o_{p_1}, ..., o_{p_n})$ is the vector of unweighted inputs to the neuron u and δ_u is defined as

$$\delta_u = \sum_{v \in U_{OUT}} (\sigma_v - \hat{\sigma}_v) \left(\frac{\partial \hat{\sigma}_v}{\partial n_u} \right), \tag{1.75}$$

where U_{OUT} is the set of neurons in the output layer of the network, $\hat{\sigma}_v$ is the output of neuron v, o_v is the corresponding groundtruth (target value), and n_u is the input of neuron u, defined as $n_u = \mathbf{w}_u \mathbf{i}_u$. If u is a neuron of the output layer, (1.75) reduces to:

$$\delta_u = (\sigma_u - \hat{\sigma}_u) \frac{\partial \hat{\sigma}_u}{\partial n_u}; \tag{1.76}$$

otherwise, if u is a hidden neuron, it becomes

$$\delta_u = \left(\sum_{s \in succ(u)} \delta_s w_{su} \right) \frac{\partial \hat{\sigma}_u}{\partial n_u}, \tag{1.77}$$

where $succ(u)$ is the set of successors of u, i.e. all the neurons in the next layer adjacent to the one which u belongs to. Therefore (1.76) and (1.77) give us the base case and the rule for the recursive computation of δ_u, which in turn allows to compute the update of each parameter inside the network.

Notice that we can express $\hat{\sigma}_u$ as $\hat{\sigma}_u = g(n_u)$, where $g(\cdot)$ is the neuron activation function in (1.71) and the term $\frac{\partial \hat{\sigma}_u}{\partial n_u}$ becomes $\frac{\partial g}{\partial n_u} = g'(n_u)$. Therefore, the backpropagation algorithm requires the activation function to be differentiable. This is the case of the Sigmoid, defined in (1.62), and of the ReLu function, defined in (1.72), except for $x = 0$, but given that this function is piece-wise linear, it actually does not represent a problem in practice. All the formulas reported above have been derived from [67].

1.3.3 Convolutional Neural Networks

The first CNN proposed was the so-called LeNet5 for document recognition by Yann LeCun *et al.* in 1994 [65]. It represents the baseline for most of the recent architectures and a true inspiration for many people in the field of DL. Many other CNNs are well known in the literature such as AlexNet [68], GoogleNet [69], ResNet [14] and VGG [70].

CNNs have been introduced to deal with image processing applications, where the huge amount of pixels make it impossible to manage images with classical ANNs, i.e. the Feed-forward Neural Networks described in the previous section. ANNs indeed tend to struggle with the computational complexity required when dealing with image data. Common ML datasets such as the MNIST database of handwritten digits, are suitable for most ANN architectures, due to their relatively small image dimensions of just 28×28. With this type of datasets, a single neuron in the first hidden layer will contain 784 weights, which is manageable for most ANN architectures. However, considering a larger RGB image as input (for example 64×64 images with three color channels), the number of weights on just a single neuron of the first layer will dramatically increase, being proportional to 10^4. It has also to be taken into account that when dealing with this input sizes, the network will need to be very large and several drawbacks will arise.

The idea at the basis of CNNs was thus to apply convolution filters to extract the semantic of input data images and construct an encoded representation through latent vectors obtained by a sequence of convolution operations. Now the network weights are actually the filters coefficients and, as a result, the network itself becomes no more dependent on image dimensions and more robust to image transformations. This allows to encode image-specific features into the architecture, making the network more suitable for image-focused tasks, incredibly reducing the parameters required to set up the model.

A way to represent the structure of a CNN is to view layers as organised along three dimensions, which are the input image dimensions (height and width) and the depth. The latter is also called the **channel** dimension and it depends on the number of convolution filters used. Unlike standard ANNs, in CNNs neurons within a given layer will be connected only to a small region of the layer preceding it. Recalling the example of 64×64 RGB images, in this case the input volume will have size $64 \times 64 \times 3$ (height, width and number of channels, respectively), leading to a final output layer with size $1 \times 1 \times n$, where n represents the possible number of classes, if a n-classes image classification problem has to be dealt with. The full input size has does be compressed into a smaller final representation, i.e. a volume of class scores stacked across the depth dimension.

CNNs result from the combination of three main operations, structured as layers:

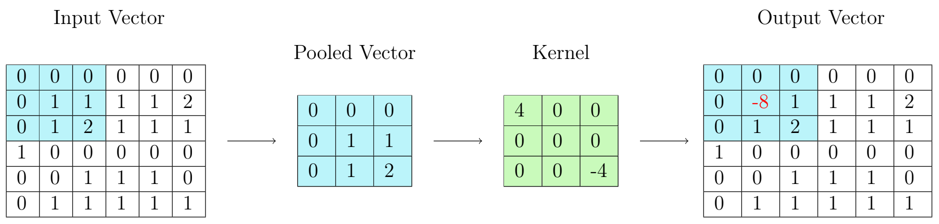


Figure 1.18: A visual representation of a convolution operation.

- **Convolution Layer:**

  As the name suggests, convolution layers play a vital role in CNNs. They are based on the use of convolution filters, also called **kernels**, whose parameters represent actually the learnable weights of the model. These kernels are usually quite smaller than the input spatial dimensions, while spread along the whole depth of the input itself. When data hit a convolution layer, the layer convolves each kernel across the input dimensions, producing a 2D activation map. When performing convolution, the scalar product between input pixels and kernel is computed, as shown in Fig. 1.18. From the resulting maps, the CNN will learn kernels that resonate with a specific feature at a given spatial position of the input. These are commonly known as activations.

  When performing convolution, three hyperparameters can be chosen, i.e. **depth**, **stride** and **padding**.

  Depth is the third dimension of the output volume produced by a convolution layer, and it can be manually set through the number of kernels applied on the input feature map. Reducing this hyperparameter can significantly minimise the total number of neurons inside the network, but it can also significantly reduce the ability of the model to detect patterns.

  Stride is the spatial step used to move the kernel along the input data between a convolution operation and the next one. Assuming we were setting a stride equal to 1, then we would have a heavily overlapped receptive field producing extremely large activations. On the other hand, setting the stride to a greater number will reduce the amount of overlapping and it will produce an output with lower spatial dimensions.

  Padding is the simple process of adding some values on the borders of the input data, and it is an effective method to further control the output volume size. CNNs have indeed the peculiarity of compressing data along successive layers. Through padding, it is possible to alter the spatial dimensions of the outputs returned by convolution layers, as

$$\frac{(V-R)+2Z}{S+1}, \tag{1.78}$$

Where V represents the input volume size, R is the receptive field size, Z is the amount of padding set and S refers to the stride. One of the most used sort of padding is **zero-padding**, that, as the name suggest, pads the input data with only zero values.

- **Pooling Layer**

  Pooling layers aim to gradually reduce the dimensions of the representation, thus further decreasing the number of parameters and the amount of computational complexity involved. A pooling layer operates over each channel of the input, and scales its spatial dimensions using a specific function, depending on the type of pooling chosen. Typical functions used are the max and the average. In the first case, we talk about **Max Pooling**, while in the second case we have **Average Pooling**. In most CNNs Max Pooling layers are used, with 2×2 kernels and applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25% of the original size, still maintaining the depth volume to its standard size.

- **Fully-connected layer**

  It contains neurons directly connected with each other through adjacent layers. This is analogous to how perceptrons are arranged in traditional ANNs as the one shown in Fig. 1.16. This network is generally positioned at the end of CNNs to give as output some compressed features used for regression and classification problems. As an example, in the problem of digit recognition [71], a CNN is used where the final layer is fully-connected with a number of neurons equal to the number of digits (classes) to be recognized. Each neuron correspond to a specific digit class, and it will contain the probability that the image given as input to the network is a graphical representation of that specific digit class. A fully-connected layer is thus fundamental when dealing with multiple-class image classification problems using CNNs.

The union of these three layers give as a result a CNN architecture, as depicted in Fig. 1.19.

1.3.4 Introduction to U-net architecture

The U-net is a CNN architecture introduced for the first time in [22] for biomedical image segmentation. This model was realized with the main purpose of achieving good performances even with a relatively small dataset available. Simpler CNNs require training procedures on many thousands of samples in order to achieve acceptable results, while the U-net, which strongly relies on data augmentation, has the peculiarity of being able to use the available samples more efficiently. U-net belongs to

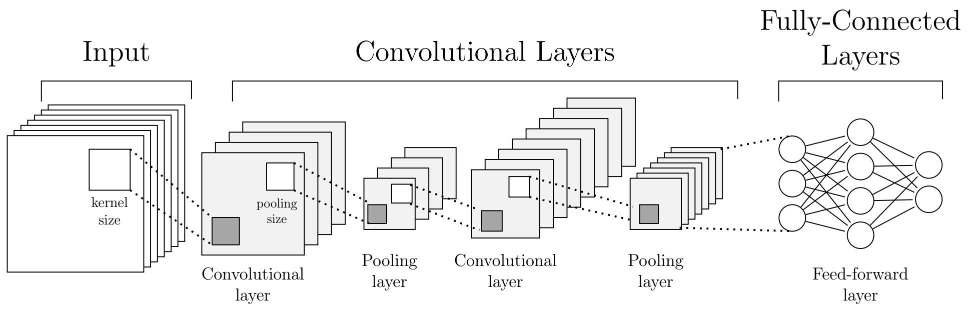


Figure 1.19: Generic architecture of a CNN

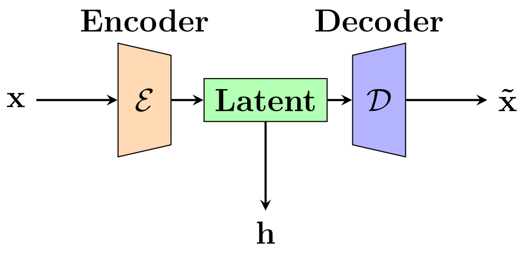


Figure 1.20: Scheme of a CA architecture.

the family of CAs, which are CNNs whose architecture can be logically split into two separate components:

- **Encoder**: maps the input $\mathbf{x}$ into the so-called hidden (or latent) representation $\mathbf{h} = \mathcal{E}(\mathbf{x})$, which is the innermost encoding layer of the autoencoder, compressing the input $\mathbf{x}$ into a lower-dimensional representation [72];

- **Decoder**: transforms the hidden representation into an estimate of the input $\tilde{\mathbf{x}} = \mathcal{D}(\mathbf{h})$.

The CA structure is sketched in Fig. 1.20. For image processing tasks as inpainting and denoising [15, 73], CAs proved to be a very powerful methodology. The U-net architecture exploits the general topology of a general CA and it is composed of three main parts:

- **Contracting/Downsampling path**

  It consists in the repeated application of convolution layers. Each of these layers will be characterized by a specific kernel size, zero-padding or not, and a final activation function. Typically ReLU is the most used one. After convolution layers, a pooling layer is defined, with specific definition of pooling size and stride. At each downsampling step, the number of feature channels is doubled.

- **Bottleneck**

  This part of the network is in between the contracting and the expanding path and it is where an encoded representation associated

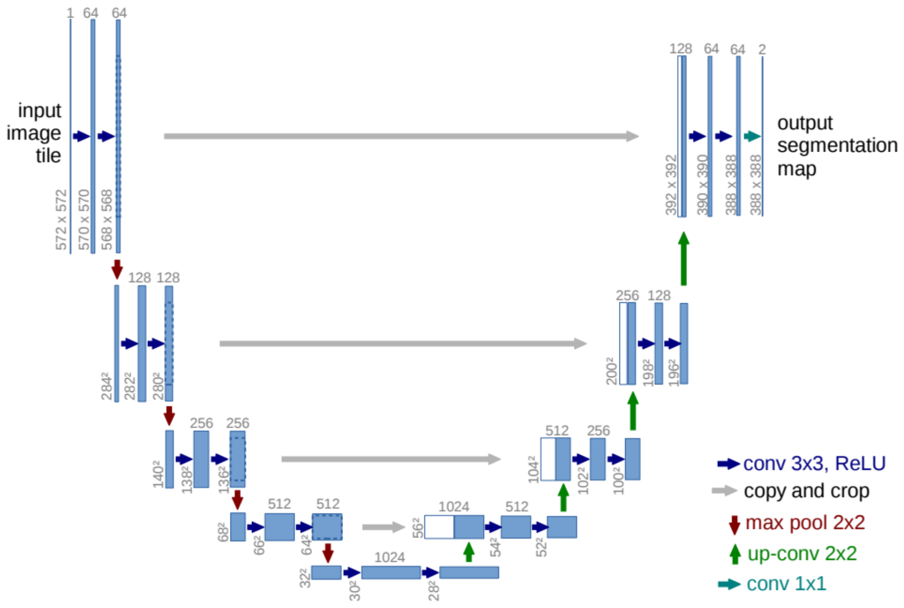


Figure 1.21: U-net architecture

to the input data, i.e. a latent vector, is built. It is typically composed by the series of two convolution layers, and usually Batch Normalization (BN) is also present. The latter is a regularization technique widely used when defining CNNs. It will better described in Sec. 1.3.5.

- **Expanding/Upsampling path**

  Every step in the expansive path consists in an upsampling of the feature map, followed by a series of convolution layers, each with a kernel size that is usually equal to that one used in the corresponding downsampling block, and a specific activation function, typically ReLU. At each upsampling step, the number of feature channels is halved. At the final layer a 1×1 convolution is usually used to map each feature vector to the desired number of classes.

Another peculiarity of the U-net is the presence of **skip connections**, which are actually connections between pairs of downsampling and upsampling blocks. These represent one of the advantages of the U-net architecture, because they combine local information from the downsampling path to the one extracted in the corresponding upsampling path, to finally obtain a general information about localisation and context necessary to predict a good segmentation map.

In Fig. 1.21 is shown a particular implementation of U-net architecture dealing with 572×572 images. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The input size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

1.3.5 Regularization Techniques

One of the greatest issues in ML models, as previously stated in Sec. 1.3.1, is overfitting. This problem usually arises when the model contains an excessive number of parameters with respect to the available data, thus learning to fit perfectly the particular realizations seen within the training set, but completely loosing its efficiency when tested on data never seen during training. In this scenario, what happens is that the model provides a low training error but high validation and test errors at the same time. In other terms, it can also be stated that the model has limited generalization capabilities. There are several techniques in order to avoid overfitting, which go under the name of regularization techniques. The approaches are various and different, so we present here the most used ones.

- **Early stopping** When training a model, training and validation errors usually have a similar decreasing trend for a certain number of epochs, but after a while the validation error may start to increase, while the training error continues to decrease. This is a hint that the model is overfitting, and also of the fact that the model itself, after having extracted some regular patterns present in the input data, is starting to learn also the noise contained in the specific realizations. We actually want to stop training at this point, where the validation error has reached its minimum. The Early stopping mechanism monitors the validation loss, or another chosen metric, and interrupts the training when it has stopped improving.

- **Dropout** In order to reduce the chance of overfitting, the weights associated to a random fraction of hidden units is set equal to 0 at each update during training. At each stage, neurons are kept in the model with a determined probability p or dropped with a probability $1 - p$, together with all their incoming and outgoing connections. Applying this regularization technique we obtain a reduced network corresponding to a simpler model involved, which helps to avoid the overfitting.

- **Batch normalization** BN was first introduced in [74] to solve the problem of covariate shifts, i.e. changes in hidden layer input distributions when passing from one layer to another, caused by the update of the parameters in the previous layers. This phenomenon slows down the learning process, requiring lower learning rates and a careful initialization of the weights. The idea of BN is therefore to normalize the input of each layer, similarly to what is usually done during the data pre-processing. A BN layer estimates the mean and the variance of each batch and uses them to perform a normalization. In order to obtain the layer output, the normalized inputs are then scaled and shifted by the two parameters. These are learned during the training process along with the network parameters and they refer to the mean and the variance. Thanks to the adoption of BN, we can speed up the learning process by increasing the learning rate. In addition, BN increases the model robustness to weights initialization, as it improves the stability of the training process.It has also a regularization effects on the model itself, due to the addition of randomness given by the batch statistics.

In this work we also used **Adam optimizer** [75]. Adam, name deriving from adaptive moment estimation, is an optimization algorithm that can be used as an extension to the classical stochastic gradient descent procedure (Sec. 1.3.1) to update network weights during training by adapting the learning rate instead of keeping it fixed. Adam derived from the combination of the advantages given by two other extensions of stochastic gradient descent, which are Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) [66]. Instead of adapting the parameter learning rates based only on the average first moment of the loss gradients (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). Specifically, the algorithm calculates an exponential moving average both of the gradient and of the squared gradient, and two parameters β_1 and β_2 are defined to control the decay rates of these moving averages. Adam is actually one of the most used optimization algorithms, particularly in DL applications to the field of image processing and computer vision.

2 Data-driven models for super-resolution

In this chapter, we present the data-driven models developed in this work for FRF signals interpolation. During the studies conducted, two U-nets architectures have been developed in parallel: the first one managing 2D space-frequency images as input-output samples; the second one working with 3D FRF tensors. Here we will give at first a formal description of the problem we want to deal with, then moving to a deeper view on the implementation of both the architectures, focusing on the layers used and on some design choices taken to make the networks more efficient in the prediction phase. In the last section of this chapter, we will discuss one of the most central points of our work, i.e. the definition of a custom loss function weighted by a specific prior mask, defined in order for our architectures to be able to fit specific FRFs features.

2.1 Problem Formulation

In our work we are going to consider a plate with a specific combination of width (i.e plate length along the x dimension), height (i.e. plate length along the y dimension) and thickness (i.e plate length along the z dimension), subjected to harmonic excitation. The second condition belonging to Kirchoff-Love assumptions (Sec. 1.1.2) is taken as valid, considering a thin plate as case study, having thickness that can be ignored if compared to the other two spatial dimensions. The plate itself can thus be approximated as a 2D system.

An harmonic excitation load, namely

$$\tilde{F}_k(\omega) = F_{k0}e^{j\omega t} \quad [\mathrm{N}], \tag{2.1}$$

is applied normally on the plate, i.e. along the z direction, in correspondence of a fixed point on the plate surface. As already deeply discussed in Sec. 1.1.3, if the carrier frequency ω of the the harmonic load $\tilde{F}_k$ applied on the plate is swept over a specific range, the plate response, that in our case is expressed in terms of normal displacement $w(x, y, \omega)$, could be acquired ideally in correspondence of any point (x, y) on the plate surface and for each value of the load carrier frequency ω in the range. From those acquired responses, the overall complex-valued frequency response H can be derived as

$$H(x, y, \omega) = \frac{w(x, y, \omega)}{\tilde{F}(\omega)}, \tag{2.2}$$

where $w(x, y, \omega)$ is the frequency counterpart of the plate normal displacement recorded in a specific spatial location (x, y). $H(x, y, \omega)$ is thus ideally continuous both in space and frequency.

However, what happens in real applications is that the system under analysis, in our case a plate, is spatially sampled using sensors to record the excitation responses. Also frequency values over which signals are recorder are not continuous, as the excitation load carrier frequency is swept over a finite range and with a finite frequency step. In other terms, a discretized version of $H(x, y, \omega)$ can be obtained by sampling on a regular spatial grid of $M \times N$ points over the surface and considering K bins equally spaced in the frequency range $[0, f_{\mathrm{MAX}}]$ (temporal frequency $f = \omega/2\pi$). The step between two consecutive frequency bins, i.e. the frequency resolution, is proportional to reciprocal of the time length of the excitation responses acquired in correspondence of the spatial sampled positions. The Nyquist-Shannon theorem, already introduced in Sec. 1.2, strictly relates the spatial sampling frequency f_s, i.e. the frequency through which the plate surface is sampled, to the highest frequency detectable f_{MAX} in the relation formalized by (1.53). The sampling operation, performed both in spatial and frequency domains, leads to the three-dimensional tensor $\mathbf{H}^{(\mathrm{HR})} \in \mathcal{R}^{N \times M \times K}$, with elements

$$[\mathbf{H}^{(\mathrm{HR})}]_{i,j,k} = |H(i\bar{x}, j\bar{y}, k\bar{\psi})|, \tag{2.3}$$

where $\bar{x}$, $\bar{y}$ are the spatial sampling steps along the x and y axes, respectively, and $\bar{\psi}$ is the sampling step in frequency.

For each point $(i\bar{x}, j\bar{y})$ on the $M \times N$ spatial grid, $\mathbf{H}^{(\mathrm{HR})}$ will contain what we call a *FRF signal*, i.e. a K-samples long signal representing the FRF magnitude evaluated in correspondence of $(i\bar{x}, j\bar{y})$ for each frequency value in the range $[0, f_{\mathrm{MAX}}]$. By fixing a sample $\hat{\jmath}$ on the y axis, a xf slice of $\mathbf{H}^{(\mathrm{HR})}$ will thus be obtained, i.e a $M \times K$ grid containing the FRF signals for each point $(i\bar{x}, \hat{\jmath}\bar{y})$ on the plate surface. By fixing a

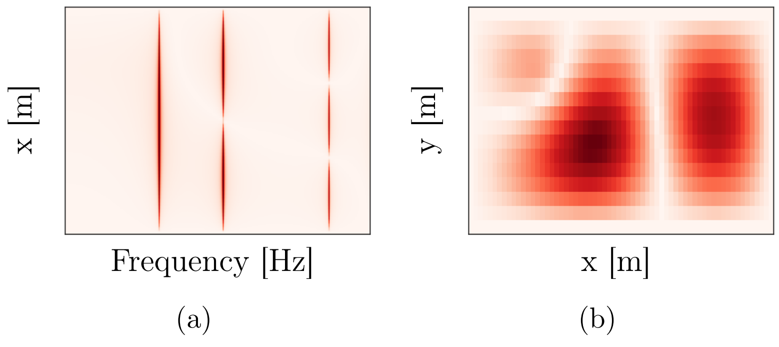


Figure 2.1: Slices from a 3D FRF tensor $\mathbf{H}^{(\mathrm{HR})}$. In (a) an example of space-frequency image obtained by slicing a tensor $\mathbf{H}^{(\mathbf{HR})}$ along the y axis. In (b) a spatial image obtained by slicing the same tensor along the the frequency axis.

sample $\hat{\imath}$ on the x axis instead, a yf slice of $\mathbf{H}^{(\mathrm{HR})}$ will be obtained, i.e a $N \times K$ grid containing the FRF signals for each point $(\hat{\imath}\bar{x}, j\bar{y})$ on the plate surface. In both cases we talk about 2D **space-frequency slices** (see Fig. 2.1a). On the other hand, fixed a frequency sample $\hat{k}$, we will obtain a $M \times N$ grid containing the FRF signals evaluated for all the spatial points on the plate sampled surface, in correspondence of the frequency value $\hat{k}\bar{\psi}$. In this case we talk about 2D **spatial slices** (see Fig. 2.1b).

In this work, we are interested in generating the tensor $\mathbf{H}^{(\mathrm{HR})}$, where FRF data are represented on a dense and regular spatial grid having dimensions $M \times N$, starting from a low-resolution version where few FRFs are available, irregularly distributed over the plate surface. Let us denote with $\mathcal{M}$ the set of locations with known FRFs, such that $|\mathcal{M}| < M \times N$. The set of signals can thus be rearranged in a tensor $\mathbf{H}^{(\mathrm{LR})} \in \mathcal{R}^{N \times M \times K}$, with elements

$$[\mathbf{H}^{(\mathrm{LR})}]_{i,j,k} = \begin{cases} |H(i\bar{x}, j\bar{y}, k\bar{\psi})|, & (i\bar{x}, j\bar{y}) \in \mathcal{M} \\ 0, & otherwise \end{cases}. \tag{2.4}$$

where $|\cdot|$ is introduced as in our work we are interesting in dealing with FRF magnitudes. Therefore, our goal is the estimation of a discrete interpolation scheme $\mathcal{F}$ such that

$$\mathbf{H}^{(\mathrm{HR})} \approx \mathcal{F}(\mathbf{H}^{(\mathrm{LR})}), \tag{2.5}$$

where $\mathbf{H}^{(\mathrm{LR})}$ is available from measurements.

Actually, the first approach we propose here is to define an interpolation scheme managing FRF data not directly stored in the 3D tensors so far described, but instead investigating the interpolation problem with 2D xf space-frequency slices taken from those tensors. Once explored this, the next step will be to define a model managing directly the 3D tensors for interpolation purposes.

Here we thus propose to implement $\mathcal{F}$, both for 2D and 3D data structures, with two versions of a specific CA, i.e. the U-net. In the field of image processing, CAs proved to be a very powerful instrument for inverse problems, thanks to their encoder-decoder architecture that confers them the ability to extract relevant features from input data. Their recent application to mode shape super-resolution in [20] has shown promising results and made us confident in the power of CAs if applied to vibrometric data interpolation. Moreover, U-net could reach in our opinion even better results in this field of analysis, thanks also to the presence of skip connections that improve the decoding phase, accounting on the features extracted by the encoder.

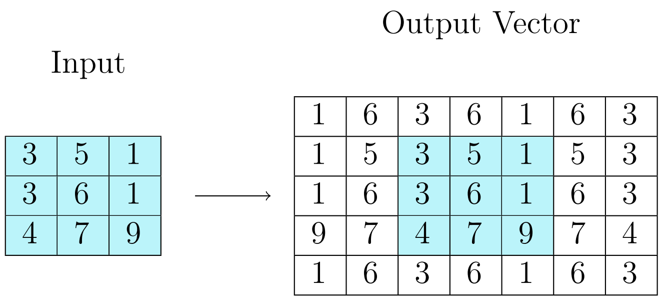
Our U-nets will be trained on simulations performed on a thin plate FEM model, in order to interpolate FRF magnitude data, where FRF is intended here in terms of receptance, i.e. ratio between plate normal displacement and excitation force. A custom loss function will be used to emphasize the correct reconstruction of specific features in the FRF magnitude (Sec. 2.3).

Notice that we are assuming to work only with the magnitude of H, but the same methodology can be adopted for the interpolation of the phase with the definition of a suitable prior inside the loss function.

2.2 Proposed architectures

We have already discussed in Sec. 1.3.4 the hierarchical structure of a U-net. Before describing the specific implementations done in this work, we want to point out some choices on one of the operations performed when applying a convolution kernel in a CNN, i.e. padding. CNNs have indeed the peculiarity of reducing the spatial size (i.e. all the dimensions apart from the channel one) of the input data for every convolutional layer data pass through, generating feature maps as output which result to be compressed representations of the inputs and whose dimensions are strictly related to the shape of the convolutional kernels adopted in relation to the input data dimensions. In other words, convolutional layers extract information from input data by compressing them and the higher is the compression, the lower will be the number of layers that can be concatenated to form the network. High compression and a consequently low number of convolutional layers concatenated, could result into an ineffective model that fails in extracting the most relevant features from input data. Moreover, when performing convolution operations, the information in correspondence of the input spatial edges is missed, and this is actually an issue, as that information could be relevant.

Padding is thus used in order to compensate the compressing effect of a convolutional layer, by the addition of rows and columns of values along the input data structure, obtaining an output feature map which is less compressed in spatial dimensions and preserving information on the input edges. As already mentioned in Sec. 1.3.3, the most commonly used form of padding is zero-padding, where the values added to input data are all zeros. This specific padding procedure has the advantages proper of



3	5	1
3	6	1
4	7	9

1	6	3	6	1	6	3
1	5	3	5	1	5	3
1	6	3	6	1	6	3
9	7	4	7	9	7	4
1	6	3	6	1	6	3

Figure 2.2: Reflection padding operation. On the right the input data and on the left the same data after $(1, 2)$ padding.

padding, thus allowing to earn valuable information from input borders, and is computationally efficient at the same time, as the added zero-value input units do not contribute to the forward pass and their corresponding synaptic weights will not be updated during the backpropagation. On the other hand, zero-padding clearly works by adding completely unrelated data to the input and this could lead to undesirable effects on the model quality.

There are other types of padding, such as reflection padding, which attempts to pad with plausible data values by re-using what is along the borders of the input. Reflection padding specifically acts by adding data which are the reflection of the input values with respect to the border axes. In Fig. 2.2 an example of reflection padding is shown, where the input is considered for simplicity as a 3×3 image having only one channel. In this example a $(1, 2)$ reflection padding is applied, meaning that the input is padded with one row per side and two columns per side. In our specific case study, we found that both 3D tensors $\mathbf{H}^{(\mathrm{HR})}$ and their 2D space-frequency slices were characterized by low dynamics, meaning that data where really sparse, with the presence of a huge amount of zero values inside. In Fig. 2.3 it can be seen the pixel intensity histogram of a space-frequency slice normalized with respect to its intensity maximum. It is noticeable that the image dynamics is really poor, with near to $6 \cdot 10^4$ occurrences of zero values. We thus decided to not use zero-padding in the implementation of convolutional layers, given that it would mean to add other zero values on data which, as demonstrated, are already sparse. We opted for reflection padding, by implementing a custom Reflection Padding layer to be inserted in our architecture both in the downsampling and in the upsampling path, with the purpose of performing $(1, 1)$ reflection padding. We used it both when dealing with 3D FRFs tensors and with 2D space-frequency slices.

A first network has been implemented to tackle a 2D version of the problem formulated in Sec. 2.1. In this case, training and testing were not considered on $(\mathbf{H}^{(\mathrm{LR})}, \mathbf{H}^{(\mathrm{HR})})$ couples directly, but on 2D xf space-frequency images obtained by slicing those tensors along the y axis. The three parts that compose the U-net architecture have been implemented

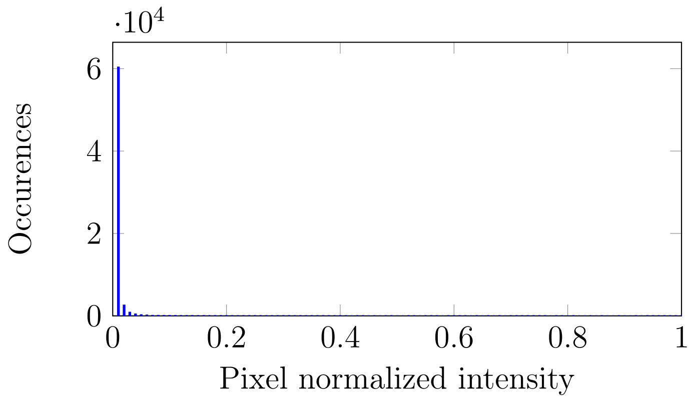


Figure 2.3: Intensity histogram of a normalized space-frequency slice.

as follows:

Downsampling path: Input data are progressively compressed in order to obtain an encoded representation of them (latent vector). In our specific application, the encoding phase should exploit the purpose of extracting and understanding some spatial patterns present in the input FRF data. As the implementation concerns, the downsampling path includes four compressing blocks, each composed by two 2D Convolution layers (Conv2D) with 3×3 filter size, stride 1×1 and ReLu activation. Each Conv2D layer is preceded by a Reflection Padding Layer [76], whose purpose has been already discussed before, and followed by Batch Normalization (BN). The number of filters C doubles by going down from one compressing block to the next, starting from $C = 16$ up to $C = 128$. While the depth of the representation increases, the spatial dimensions of the images are halved at each step by a Max Pooling layer with pool size 2×2, positioned at the end of each downsampling layer.

Bottleneck: This part of the network should provide a hidden representation of dense and clean FRF images, that should be effective independently from the local spatial patterns and from the specific image given as input. It should comprehend the dynamics that are proper of vibrometric space-frequency images. In our implementation, it is composed by a series of two Conv2D layers with 3×3 filter size, stride 1×1 and ReLu activation [77], each one followed by a BN layer. The number of filters used is $C = 256$.

Upsampling path: The decoding phase should implement a kind of non linear interpolation scheme of FRF data, starting from the knowledge gained in the encoding phase, in order to have at the end a dense and regularly sampled space-frequency image. The expanding phase is symmetrical with respect to the encoder, with four upsampling blocks, each composed by two Conv2D with 3×3 filter size, stride 1×1 and ReLu activation. Each Conv2D layer is preceded by a custom Reflection Padding layer, and followed by BN. The number of filters C of the two Convolution layers halves by going up from one upsampling block to the next, starting from $C = 128$ down to $C = 16$. While the depth of the representation decreases, the spatial dimensions of the images are doubled at each step by a 2D UpSampling layer (Up2D) with upsampling factors 2×2, positioned at the end of each upsampling layer. This UpSampling layer performs an upscaling of the input image, through a nearest neighbour interpolation scheme.

Skip connections [78] are added between each pair of corresponding downsampling and upsampling blocks, in order to reuse the low resolution features gathered in the encoder during the upsampling phase.

A final 2D Convolution layer is then introduced, characterized by one filter, kernel size 1×1, zero-padding and Sigmoid activation.

The introduction of batch normalization after convolutional layers particularly improved the U-net performances. In the original model indeed, BN layers were not present and, as a result, our model itself had the tendency to get stuck on a local minimum reached just after the first epoch of training. The reconstructions provided were actually composed by only zero values, and this in our opinion was due to the fact that, as input data are really sparse, the model found as best solution that one of filling the missing data with all zero values, failing completely the reconstruction. Instead of acting by performing pre-processing histogram equalization on input data to be learnt by the model, we thus opted for inserting batch normalization inside the model itself. Batch normalization, by normalizing layer inputs, demonstrated to give a fundamental contribution to the attenuation of sparsity of data, performing histograms equalization directly inside the model.

We also defined a 3D U-net architecture in order to deal with 3D tensors input-output couples $(\mathbf{H}^{(\mathrm{LR})}, \mathbf{H}^{(\mathrm{HR})})$. In this case the U-net manages data structures containing FRFs measurements determined over the 2D space, not limiting the analysis to only one spatial dimension, as done instead by the 2D U-net when dealing with space-frequency images. As already seen in Sec. 2.1, the proper representation of FRF measurements is through 3D tensors $\mathbf{H}^{(\mathrm{HR})}$, so for us, having an architecture able to manage directly those 3D data structure represents a step forward with respect to the 2D architecture. The extension of the operator $\mathcal{F}$ to the other spatial dimension has indeed the potential advantage of exploiting information on the overall spatial dependency of the system response, having at the end an interpolation scheme able to generate the whole spatial sampling grid in one prediction instead of one dimension at a time, which is a desirable feature in EMA. As implementation is concerned, the 3D U-net has been defined with the same architecture of the 2D one. The only difference concerns the input dimensions, that in this case are three-dimensional. Both the U-net architectures have been implemented in Keras [79]. In Fig. 2.4 it can be seen a generic representation of the proposed architectures, where tensors are represented also as sets of 2D xf space-frequency images.

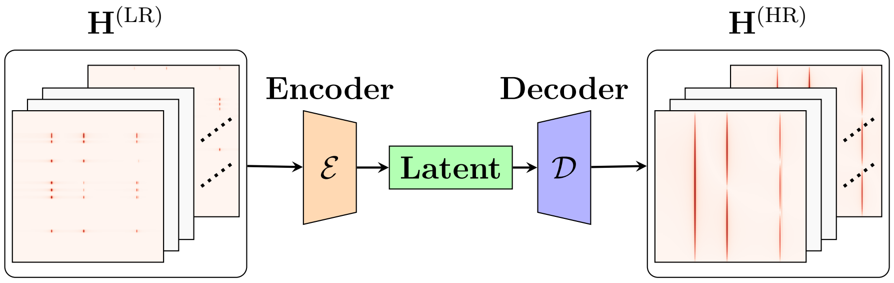


Figure 2.4: Schematic representation of the proposed architecture. The 2D U-net is trained on space-frequency images obtained slicing $\mathbf{H}^{(\mathrm{LR})}$ and $\mathbf{H}^{(\mathrm{HR})}$, while the 3D U-net works directly on the 3D tensors.

2.3 Loss Function

Data interpolation for super-resolution can be addressed as a regression problem, as input and target output belongs to the same domain, either that one of 2D images or the one of 3D tensors. In Sec. 1.3.1 we already said that the classical loss function used for regression problems is the *Mean Squared Error* (MSE) between targets and predictions [80, 81]. It has already been formalized in (1.64) and now we can give a new formulation of it, properly contextualized to our case study:

$$MSE(\mathbf{T_k}, \hat{\mathbf{T}}_\mathbf{k}) = ||\mathbf{T_k} - \hat{\mathbf{T}}_\mathbf{k}||_2^2, \tag{2.6}$$

where $||\cdot||_2^2$ is the L2 norm, $\mathbf{T_k}$ is the k_{th} target tensor in the dataset $\mathcal{D}$, and $\hat{\mathbf{T}}_\mathbf{k}$ is the corresponding prediction. $\mathbf{T_k}$ can be either the whole 3D tensor $\mathbf{H}^{(\mathrm{HR})}$, or a space-frequency image obtained by slicing $\mathbf{H}^{(\mathrm{HR})}$ along the y axis.

Actually, in our work we have not used the MSE just in its standard formulation, but we decided to apply a mask $\mathbf{M}$ imposing prior weights in the computation of the loss [82]. In image processing tasks, using prior weight masks for the computation of the MSE, helps in making an a-priori decision on the regions where we want the loss to be reduced more. In our case study, where we are dealing with FRF data, we are interested in retrieving information particularly in correspondence of the resonant peaks, whose amplitude and width are strictly related to the modal parameters of the system of reference and thus on its vibration properties. Another point of interest is the dynamic behaviour in correspondence of the spatial edges, as it is strictly related to the boundary conditions imposed. We had the opportunity to observe that both our U-nets resulted in poor predictions when trained with the standard formulation of the MSE shown in (2.6). This is again one of the issues deriving from sparsity of the FRF data we are dealing with. All these considerations and experiences convinced us in the use of a proper weight mask $\mathbf{M}$, thus defining a custom MSE as

$$MSE_{mask}(\mathbf{T_k}, \hat{\mathbf{T}}_\mathbf{k}) = ||(\mathbf{T_k} - \hat{\mathbf{T}}_\mathbf{k}) \otimes \mathbf{M}||_2^2, \tag{2.7}$$

where $\otimes$ is the Hadamard product. The optimal model weights are thus estimated as

$$\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{\mathbf{T_k} \in \mathcal{D}} MSE_{mask}(\mathbf{T_k}, \hat{\mathbf{T}}_\mathbf{k}). \tag{2.8}$$

We defined the weighting mask $\mathbf{M}$ in order to preserve three specific features of the FRFs in the training data. A first mask

$$\mathbf{M_1} = \mathbf{T_k} \tag{2.9}$$

has the purpose of improving the reconstruction in correspondence of the FRFs magnitude maxima, i.e. the resonances of the system. A second mask $\mathbf{M_2}$, defined as

$$\mathbf{M_2} = \max(\mathbf{M_1}) - \mathbf{M_1}, \tag{2.10}$$

has been defined to improve the reconstruction at the antiresonances. This mask in particular has been introduced with the main purpose of attenuating the effects of $\mathbf{M_1}$, preventing that an improved reconstruction obtained in correspondence of the resonant peaks thanks to $\mathbf{M_1}$, could lead to a particularly bad reconstruction of the antiresonances. In other words $\mathbf{M_2}$, if properly combined with $\mathbf{M_1}$, helps in preserving still good reconstructions of those areas having low dynamics.

A third mask $\mathbf{M_3}$ has been defined to impose a prior on the spatial edges. When dealing with 3D tensors it has the following form

$$[\mathbf{M_3}]_{i,j,k} = \begin{cases} C \textit{ if i=0} \cup \textit{i=M-1} \\ C \textit{ if j=0} \cup \textit{j=N-1} \\ 0 \textit{ otherwise} \end{cases}, \tag{2.11}$$

while in case of dealing with 2D xf space-frequency slices, $\mathbf{M_3}$ changes to

$$[\mathbf{M_3}]_{i,k} = \begin{cases} C \textit{ if i=0} \cup \textit{i=M-1} \\ 0 \textit{ otherwise} \end{cases}, \tag{2.12}$$

as in this latter case, only one spatial dimension is accounted for, i.e. the x axis. C is a constant weight to be properly chosen. We know indeed that, in the case of a system with simply supported boundaries, the edges displacement is prevent while rotation is free. On the other hand, for a system having clamped boundaries, both displacement and rotation are prevent on the spatial edges. In other words, edges dynamics are strictly related to the boundary conditions imposed, so the purpose of $\mathbf{M_3}$ is to train both the U-nets to be able to discern those conditions when performing interpolation. Finally, the overall mask $\mathbf{M}$ is a linear combination of the three priors, namely

$$\mathbf{M} = w_1 \cdot \mathbf{M_1} + w_2 \cdot \mathbf{M_2} + w_3 \cdot \mathbf{M_3}, \tag{2.13}$$

where w_1, w_2 and w_3 are the coefficients to be found through a proper hyperparameter tuning.

3 Dataset Generation

This chapter introduces the readers to the initial step of this work, i.e. the construction of the dataset needed to train and test the proposed U-net architectures. Firstly, it will be discussed how data have been obtained by introducing COMSOL Multiphyics® as the reference software used for vibrometric data synthesis, giving also an inside view of how simulations have been automated by means of MATLAB® LiveLink™. Finally, it will be described how the acquired data have been organized in different data structures using Python, to create the two final datasets, one used to train and test the 2D U-net and the other in order to do the same for the 3D U-net.

3.1 Simulations in COMSOL Multiphysics

COMSOL Multiphysics® is a simulation software based on FEM, which offers conventional physics-based user interfaces and predefined coupled systems of partial differential equations (PDEs). For the simulations to be conducted, the **Structural Mechanics Module** has been chosen. This is an optional add-on package that extends the modeling environment with customized physics interfaces designed to solve problems in the fields of structural and solid mechanics, including a special interface for modeling thin structures such as shells, membranes, beams, plates, and trusses.

In our specific case we built the model of a thin rectangular plate. In order to do so, a Work Plane has been used, inside of which we defined a rectangular planar geometry. The next step was to define the point of application of the normal load over the plate surface. A cylinder geometry has thus been defined with radius $r = 0.006\,\text{m}$, oriented normally to

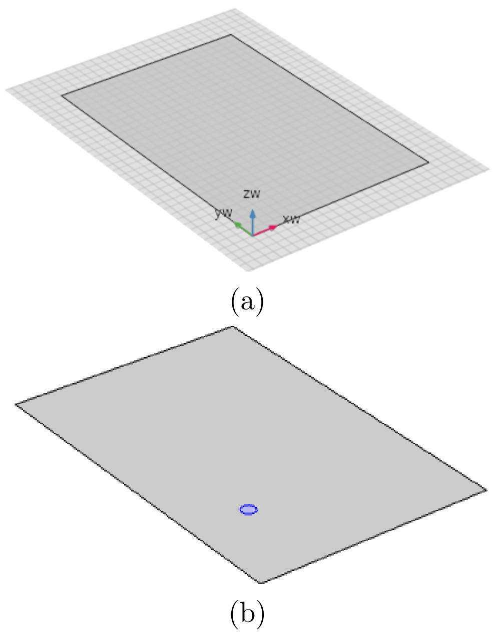


Figure 3.1: Implementation steps of the plate geometry in COMSOL. In (a) the initial plate geometry and in (b) the final geometry with a point load defined on it

the plate geometry, i.e. along the z axis, and passing through (x_L, y_L). This point belonging to the plate surface will correspond to the normal load point of application. Values associated to those coordinates are chosen with respect to the plate size and will be clarified in Sec. 3.2 Once both the plate and the cylinder geometries have been defined, an Intersection module has been used to determine the final rectangular plate geometry including the presence of the point load. In Fig. 3.1 both the original plate geometry with respect to the reference system, and the result of the intersection between the geometry itself and the cylinder are shown. By following the steps described above, the point load has actually been modelled as a closed surface lying on the plate itself. This turned out to be the best choice to perform effective vibrometric studies on the plate geometry. A first attempt was made to model the point of application of the load as an adimensional point on the plate surface, but this choice revealed some problems when performing the simulations, with the appearance of some sort of singularities when computing the solution.

As material choice was concerned, we chose to associate an isotropic and linear elastic material to the plate geometry, with Density $\rho = 370\,\mathrm{kg/m^3}$, Young's modulus $E = 10.8\,\mathrm{GPa}$ and Poisson's ratio $\nu = 0.372$.

We used the **Shell interface** to associate those elastic properties to the plate geometry, to define the type of load to be applied and also the boundary conditions to be imposed. A unitary magnitude **face load**

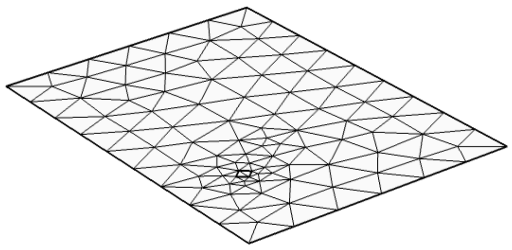


Figure 3.2: Triangular mesh built over plate surface

normal to the plate surface has thus been defined as

$$\mathbf{F}_{tot} = [0, 0, 1]^T \quad [\mathrm{N}], \tag{3.1}$$

to be applied on the boundary generated by the intersection between the rectangular plate and the cylinder, and both **Clamped** and **Simply Supported** boundary conditions have been considered. We also defined a **mesh**, i.e. a discrete set of points lying on the plate surface where the study solution had to be computed. We chose a triangular mesh, specifying also the maximum element size to be

$$\Delta r_{MAX} = \lambda_{min}/6, \tag{3.2}$$

where λ_{min} is the minimum wavelength observable in the frequency range $[0\,\mathrm{Hz}, 1024\,\mathrm{Hz}]$. In Fig. 3.2 it can been seen clearly the triangular mesh built over the plate surface. It can be also noticed that the mesh itself gets finer in correspondence of the point load and this is typically implemented by COMSOL when performing studies based on FEM. Finally, in order to obtain the FRF data associated to the plate, we chose to perform a **Frequency Domain Study**, which is used to compute the response of a linear or linearized model subjected to harmonic excitation for one or several frequencies. This type of study accounts for the effects of all the mode shapes that are properly resolved by the mesh built on a model subjected to loads or excitations, and its output is displayed as a transfer function, e.g. magnitude or phase of deformation, sound pressure, impedance, or scattering parameters as a function of frequency.

We used the load defined in (3.1) as excitation to be applied on the plate, and a range going from 0 Hz to 1024 Hz, with a frequency step of 1 Hz for the computation of the frequency responses. The output of the frequency domain study has been computed in terms of plate normal deformation for each value belonging to the frequency range, as shown in Fig. 3.3.

3.2 Simulations Campaign

After the plate model has been initialized, we iterated the frequency domain studies over plates with different dimensions, in order to build a sufficiently wide and diversified dataset. To do so, we used MATLAB®

LiveLink$^{\text{TM}}$, which is an extension used to connect COMSOL Multiphysics® to the MATLAB® scripting environment. This allowed us to establish an interactive modeling between the COMSOL Desktop (Sec. 3.1) and MATLAB®, sharing the same plate model. Every modification performed at the MATLAB® prompt is simultaneously updated in the COMSOL Desktop.

The plate width l_x have been varied in the range $[0.23\,\text{m}, 0.36\,\text{m}]$ with $\Delta x = 0.01\,\text{m}$, the plate height l_y in the range $[0.15\,\text{m}, 0.2\,\text{m}]$ with $\Delta y = 0.01\,\text{m}$ and finally the plate thickness t have been varied in the range $[0.001\,\text{m}, 0.01\,\text{m}]$ with $\Delta t = 0.001\,\text{m}$. Those ranges have been chosen as they represent the typical dimensions of violin top plates. The combination of all the possible geometrical dimensions gave a total of 840 plates. By varying also between the two possible boundary conditions previously defined, we ended up with a dataset of 1680 plates. A loop function has thus been defined where, at each iteration, a frequency domain study in the frequency range $[0\,\text{Hz}, 1024\,\text{Hz}]$ was performed on a plate with a specific combination of width l_x, height l_y, thickness t and boundary conditions.

The point of application of the load, necessary for the frequency domain study to be computed, was located in $\boldsymbol{r_k} = (0.08\,\text{m}, 0.05\,\text{m})$, so in proximity of the bottom-left corner of the plate geometry. This choice was taken after having performed a preliminary **Eigenfrequency Study** on the plate geometry, to see the plate mode shapes and their relative nodal lines. The study showed that the region around $\boldsymbol{r_k}$ was on average the one with less occurrences of nodal lines over the different mode shapes so, in order to excite the highest number of mode shapes possible, $\boldsymbol{r_k}$ has been chosen as the load point of application.

At each iteration of the for loop, a section dedicated to data export has been defined after the computation of the frequency domain study. For each simulation, we sampled the plate surface through a regular 2D spatial grid with dimensions $M \times N$, where $M = 64$ is the number

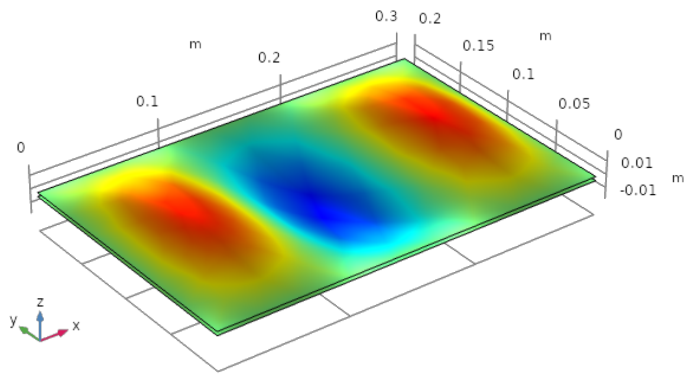


Figure 3.3: Result of the Frequency Response study on a rectangular plate analyzed at 1 kHz. Data are plotted in terms of plate's normal deformation

of samples along the plate width, considering a sampling step equal to $l_x/(M-1)$, and $N = 16$ is the number of samples along the plate height, considering a sampling step equal to $l_y/(N-1)$. We exported data in terms of plate normal deformation w at the spatial grid nodes for each frequency value in the range $[0\,\mathrm{Hz}, 1024\,\mathrm{Hz}]$, obtaining a $64 \times 16 \times 1024$ regularly sampled tensor $\mathbf{H}^{(\mathbf{HR})}$ associated to each of the 1680 plates.

3.3 Dataset Preparation

The 3D tensors $\mathbf{H}^{(\mathbf{HR})}$ already represent the targets for the 3D U-net. To prepare the corresponding model inputs $\mathbf{H}^{(\mathrm{LR})}$, some FRFs have been replaced with zero-valued vectors, according to a 3D binary mask $\mathbf{D}$, even regular or random. It is important to remark that in our work we are interested in performing a spatial downsampling procedure, that does not mean we are removing FRF data along the frequency axis, but instead it means we are removing entire FRF signals in correspondence of some specific spatial locations.

The first step of our analysis was to use a 3D regular binary mask $\mathbf{D}$, thus defining regularly downsampled input tensors $\mathbf{H}^{(\mathrm{LR})}$. Defined as

$$\mathcal{L}_{\mathbf{x}} = \{i \in \{0,1\} : \textit{ i=1 if i mod } d_x \textit{= 0}\},\ |\mathcal{L}_{\mathbf{x}}| = M, \tag{3.3}$$

$$\mathcal{L}_{\mathbf{y}} = \{j \in \{0,1\} : \textit{ j=1 if j mod } d_y \textit{= 0}\},\ |\mathcal{L}_{\mathbf{y}}| = N, \tag{3.4}$$

two binary lists representing the deterministic selection of indices (i, j) along the x and y axes in $\mathbf{H}^{(\mathrm{HR})}$ respectively, a 2D binary mask $\mathbf{B}$ can be determined as

$$\mathbf{B} = \mathcal{L}_{\mathbf{x}}\mathcal{L}_{\mathbf{y}}{}^{T}, \tag{3.5}$$

and the final 3D binary mask $\mathbf{D}$ is derived from (3.5) as

$$[\mathbf{D}]_{i,j} = \begin{cases} \mathbf{1}_{1024} & \textit{if } [\mathbf{B}]_{i,j} = 1 \\ \mathbf{0}_{1024} & \textit{otherwise} \end{cases}, \tag{3.6}$$

so that $\mathbf{H}^{(\mathrm{LR})} = \mathbf{D} \otimes \mathbf{H}^{(\mathrm{HR})}$. The values of the downsampling factors (d_x, d_y) were chosen in order to simulate different percentages of known data during the training: $(2, 1)$, $(2, 2)$ and $(4, 2)$ correspond to $50\,\%$, $25\,\%$ and $12.5\,\%$ of available data, respectively. In Fig. 3.4, a spatial slice from a regularly downsampled tensor with 25% of original data and a slice of the corresponding 3D regular binary mask $\mathbf{D}$ are shown.

The step forward was to use a 3D random mask $\mathbf{D}$ to build irregularly sampled tensors $\mathbf{H}^{(\mathrm{LR})}$ to be given as inputs to the 3D U-net. The 2D binary mask $\mathbf{B}$ is now defined as:

$$\mathbf{B} = \mathbf{b}_{\mathbf{x}}\mathbf{b}_{\mathbf{y}}{}^{T}, \tag{3.7}$$

where $\mathbf{b}_{\mathbf{x}} \sim \mathcal{B}(N, p_x)$ and $\mathbf{b}_{\mathbf{y}} \sim \mathcal{B}(M, p_y)$ are realizations of two binomial distributions representing the random selection of indices (i, j) along the

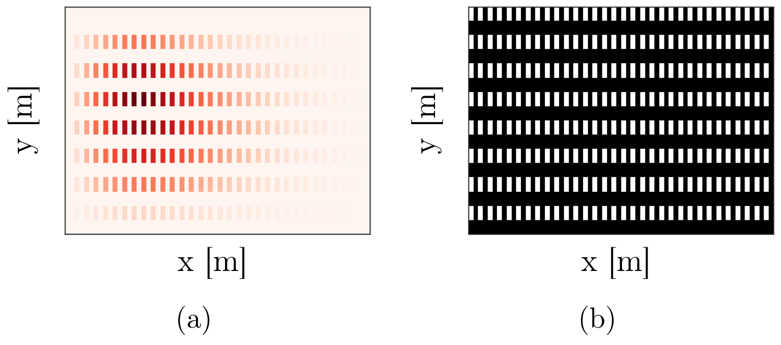


Figure 3.4: Example of spatial slice with missing data and the relative regular binary mask. In (a) the spatial slice taken from a tensor $\mathbf{H}^{(\mathrm{LR})}$ with 25% of original FRF data and in (b) a slice of the corresponding 3D regular binary mask $\mathbf{D}$

x and y axes of the sampling grid. The final 3D binary mask $\mathbf{D}$ is then derived again as in (3.6), where now $\mathbf{B}$ is given by (3.7), so that $\mathbf{H}^{(\mathrm{LR})} = \mathbf{D} \otimes \mathbf{H}^{(\mathrm{HR})}$. The values of the probabilities (p_x, p_y) are chosen in order to simulate different percentages of known data during the training: $(0.5, 1)$, $(0.5, 0.5)$ and $(0.25, 0.5)$ correspond to 50 %, 25 % and 12.5 % of available data, respectively.

Notice that, in the irregular sampling case, a new random mask $\mathbf{D}$ has been generated for each tensor, thus the network was not trained and tested on the same irregular grid. Our purpose when using random masks, was indeed that one of training our models to be able to learn features that are proper of FRF data, independently of where data were missing. This could lead to stronger models having the ability to generalize their behaviour and to provide good reconstructions when dealing with vibrometric data affected by different degrees of corruption. In Fig. 3.5, a spatial slice from a tensor with 25% of original data and a slice of the corresponding 3D random binary mask $\mathbf{D}$ are shown.

Tensors $\mathbf{H}^{(\mathrm{HR})}$ have been sliced along the y axis, obtaining a dataset of 26880 space-frequency images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{HR})}$ with shape 64×1024, used to train a preliminary model able to manage only 2D data, i.e. the 2D U-net. As done in the 3D case, the first step of our analysis was to use a regular mask $\mathbf{D}$ to prepare the model inputs $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$, thus defining regularly downsampled input images. Defined

$$\mathcal{L}_{\mathbf{x}} = \{i \in \{0,1\} : \; \mathit{i{=}1 \; if \; i \; mod \; d_x{=}\; 0}\}, \; |\mathcal{L}_{\mathbf{x}}| = M, \tag{3.8}$$

as a binary list representing the deterministic selection of indices i along the x axis in $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{HR})}$, the 2D regular binary mask $\mathbf{D}$ applied to obtained the regularly downsampled input images can be defined as

$$[\mathbf{D}]_{i,k} = \begin{cases} 1 & \mathit{if} \; [\mathcal{L}_{\mathbf{x}}]_i = 1 \\ 0 & \mathit{otherwise} \end{cases} . \tag{3.9}$$

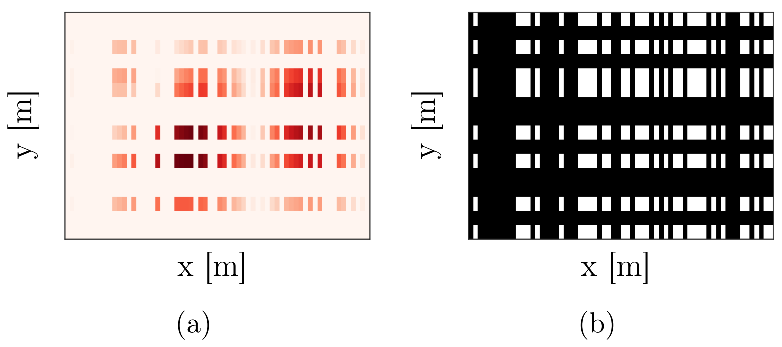


Figure 3.5: Example of spatial slice with missing data and the relative random binary mask. In (a) the spatial slice taken from a tensor $\mathbf{H}^{(\mathrm{LR})}$ with 25% of original FRF data and in (b) a slice of the corresponding 3D random binary mask $\mathbf{D}$

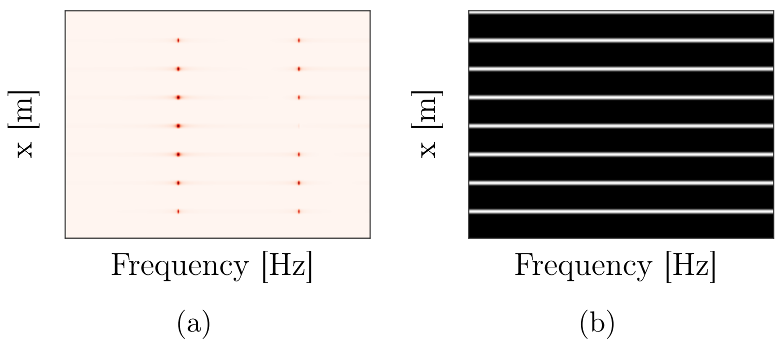


Figure 3.6: Example of a regularly downsampled space-frequency image with 12.5% of original FRF data and the relative binary mask. In (a) a regularly downsampled space-frequency image $\mathbf{I}_{\mathrm{sf}}^{(\mathrm{LR})}$. In (b) the corresponding 2D binary mask $\mathbf{D}$

thus $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})} = \mathbf{D} \otimes \mathbf{I}_{\mathrm{xf}}^{(\mathrm{HR})}$. The value of the downsampling factor d_x has been chosen in order to simulate different percentages of known data during the training: 2, 4 and 8 correspond to 50 %, 25 % and 12.5 % of available data, respectively. In Fig. 3.6 an example of a space-frequency image with 12.5% of original data is shown, along with the relative 2D regular binary mask $\mathbf{D}$. The second step was to deal with irregularly sampled slices $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ to be used as inputs of our 2D U-net. In order to prepare them, we replaced some FRFs signals with zero vectors, according to a 2D random binary mask $\mathbf{D}$. If $\mathbf{b_x} \sim \mathcal{B}(N, p_x)$ is a realization of a binomial distribution representing the random selection of indices i along the x axis in $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{HR})}$, the final 2D binary mask $\mathbf{D}$ can be defined as

$$[\mathbf{D}]_{i,k} = \begin{cases} 1 & \textit{if } [\mathbf{b_x}]_i = 1 \\ 0 & \textit{otherwise} \end{cases}, \tag{3.10}$$

thus $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})} = \mathbf{D} \otimes \mathbf{I}_{\mathrm{xf}}^{(\mathrm{HR})}$. The probability value p_x has been chosen in order

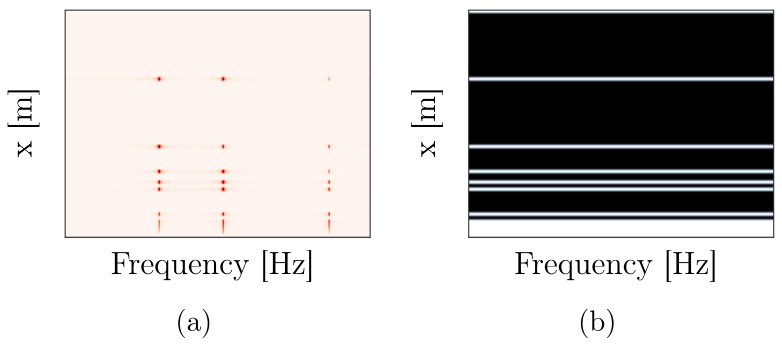


Figure 3.7: Example of a space-frequency image with 12.5% of original FRF data and the relative random binary mask. In (a) an irregularly sampled space-frequency image $\mathbf{I}_{\text{sf}}^{(\text{LR})}$. In (b) the corresponding 2D random binary mask $\mathbf{D}$

to simulate different percentages of known data during the training: 0.5, 0.25 and 0.125 correspond to 50 %, 25 % and 12.5 % of available data, respectively. Also in the 2D case, when performing irregular sampling, a new random mask $\mathbf{D}$ has been generated for each image, so the 2D U-net was not trained and tested on the same irregular grid. In Fig. 3.7 a space-frequency image with 12.5% of original data is shown, along with the relative 2D random binary mask $\mathbf{D}$.

Before training our models, both in the 2D and in the 3D case, images and tensors have been normalized with respect to their maximum, thus having data distributed between 0 and 1.

4
Results

In this chapter we summarize the results obtained by validating the models proposed in Chapter 2. We will show the performance of both 2D and 3D U-nets, with some insights on the metrics used during training and after training to qualify our models and on the test cases we have dealt with. Some visual reconstructions will be analysed both when dealing with synthetic and real-measured FRF data.

4.1 Test Cases

As already introduced in Sec 3.3, in order to prepare the model inputs, both in the 2D and 3D cases, what we have done was to apply either a regular or a random binary mask to the regularly sampled data. Our purpose indeed was to check at first the performance of our models when dealing with regularly downsampled data, i.e. 2D images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ or 3D tensors $\mathbf{H}^{(\mathrm{LR})}$ where FRFs signals are removed periodically in the spatial domain. As it can be imagined, this is actually an ideal case of study and the easiest possible our models should deal with. We also expect that in this particular scenario, the model-based interpolation schemes chosen as competitors will perform optimally, as it is well known that those algorithms are particularly efficient when interpolating on regular grids. The next step was to use instead random masks to obtain some irregularly sampled versions of both the 3D tensors and the 2D images. It is important to clarify that, both in regular and random cases, we did not remove data from the FRF signals at some frequency values in the range $[0Hz, 1024Hz]$, but we actually removed FRF signals in their entire frequency range, in correspondence of some points disposed on the $M \times N$ spatial grid. In the case of regular masks, FRFs signals have been removed following a fixed periodicity in the spatial domain, while in the random case the spatial positions where FRFs were removed have been chosen randomly. This latter case has the purpose of emulating bad experimental scenarios that could appear when performing vibrometric data acquisitions, when some specific points irregularly disposed on the system surface under study cannot be accessed for measurement and therefore entire FRFs signals cannot be obtained in correspondence of those points.

This Results section is thus organized following the logic behind our studies, which was that one of starting by dealing with ideal cases, then moving to more realistic scenarios. We will thus first analyze the results achieved by our 2D U-net when dealing with 2D space-frequency images $\mathbf{I}_{\mathrm{xf}}$ containing synthetic FRFs data obtained by FEM simulations on COMSOL Multiphyics®. At first will be investigated the performance of our model when interpolating regularly downsampled images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$, by then moving to the more realistic case of interpolating irregularly sampled images. The following section will instead be dedicated to the 3D case. Moving from images to tensors means adding information along another spatial dimension, i.e. the y axis, thus having complete representations of FRF data, expressed now both in the 2D space and in frequency. A model that manages directly those tensors, as the 3D U-net we are proposing here, interpolates over a huge amount of FRF data if compared to those contained in single space-frequency images. In our opinion, this means having a model able to achieve a deeper understanding of vibrometric data and higher capabilities to extract meaningful features. Also in this case, we will start by analyzing the performance of our 3D U-net when interpolating regularly downsampled data, by then moving to the more realistic case of irregularly sampled 3D tensors interpolation. Finally, a real case study will be considered, testing the 2D U-net as an interpolation method on an irregularly sampled 2D space-frequency image acquired through experimental measurements performed on a real thin wooden plate.

In all the test cases considered, we trained the U-net using 80% of the available dataset for training, 10% for validation and 10% for testing. Training has been performed over 20 epochs, using Adam optimizer and adjusting the learning rate, equal to 0.0004, when a plateau was detected. After a long hyperparameter tuning campaign, we found $w_1 = 0.7$, $w_2 = 0.3$, $w_3 = 1$ and $C = 0.5$ as best values for the masked MSE parameters introduced in Sec. 2.3. More precisely, we varied those parameters over a grid, checking the U-net performances in terms of masked MSE, thus finding this combination of values as the one giving the best results.

4.2 Reconstructions quality metrics

The accuracy of both the U-nets reconstructions, once training and validation procedures were already completed, has been assessed in terms of two specific metrics: the Normalized Mean Squared Error (NMSE), and the Normalized Cross Correlation (NCC). Those metrics, if contextualized to our problem, can be expressed as

$$\text{NMSE}(\mathbf{T_k}, \hat{\mathbf{T}}_\mathbf{k}) = 20log_{10}\left(\frac{||\mathbf{T_k} - \hat{\mathbf{T}}_\mathbf{k}||_2^2}{||\mathbf{T_k}||_2^2}\right), \tag{4.1}$$

$$\text{NCC}(\mathbf{T_k}, \hat{\mathbf{T}}_\mathbf{k}) = \frac{sum(\mathbf{T_k} \otimes \hat{\mathbf{T}}_\mathbf{k})}{\sqrt{||\mathbf{T_k}||_2^2 \cdot ||\hat{\mathbf{T}}_\mathbf{k}||_2^2}}, \tag{4.2}$$

computed for each couple $(\mathbf{T_k}, \hat{\mathbf{T}}_\mathbf{k})$, where $\mathbf{T_k}$ is a target sample in the test set (2688 images and 168 tensors in the 2D and 3D cases, respectively) and $\hat{\mathbf{T}}_\mathbf{k}$ is the corresponding reconstruction. Specifically, the term $sum(\mathbf{T_k} \otimes \hat{\mathbf{T}}_\mathbf{k})$ in the definition of the NCC indicates the summation over all the elements of $\mathbf{T_k} \otimes \hat{\mathbf{T}}_\mathbf{k}$. The lower the NMSE and the higher the NCC, the better the quality of the reconstruction $\hat{\mathbf{T}}_\mathbf{k}$.

In the image coding and computer vision literature, NMSE is the most frequently used measures of image quality reconstruction. The reasons for its widespread popularity is its mathematical tractability and the fact that it is often straightforward to design systems that minimize the MSE, and therefore the NMSE. We thus decided to follow this trend and use this metric to check the quality of the reconstructions provided by both our U-nets. On the other hand, it is also true that NMSE does not necessarily correspond to all aspects of the observer's visual perception of the errors. It gives indeed a measure of the distributed error, quantifying the accuracy of the provided reconstruction with respect to the target, in terms of pixel values, but it actually does not give insights into the overall reconstructed patterns. The idea behind the usage of NCC for our specific task, was thus to have a quality metric able to detect if the FRF patterns, in both tensors $\mathbf{H}$ and images $\mathbf{I}_{\text{xf}}$, where properly learnt by our models. NCC is indeed widely used as an effective similarity measure in matching tasks in the field of computer vision, as it is easy to implement and invariant to linear brightness and contrast variations. In practice, NCC has value 0 if the target $\mathbf{T_k}$ and the reconstruction $\hat{\mathbf{T}}_\mathbf{k}$ are completely uncorrelated (worst case), while it should return 1 if the target $\mathbf{T_k}$ and the reconstruction $\hat{\mathbf{T}}_\mathbf{k}$ are completely correlated, i.e. the reconstruction matches exactly the target (ideal best case).

We compared the average value and the standard deviation of both the NMSE and the NCC, analysing their behaviour with respect to different percentages of known data both during training and testing. As a baseline, we compared the results to a basic interpolation scheme with sinc functions, whose performance is known to be dependent on the amount of available data. More precisely, the interpolation scheme adopted as a competitor was a Fourier-based one. Given a downsampled input, that in our case can be either a 3D FRF tensor or a 2D space-frequency image, this algorithm works by calculating the spectrum

	50%	**25%**	**12.5%**
Train Loss	$1.5 \cdot 10^{-8}$	$2.4 \cdot 10^{-8}$	$2.5 \cdot 10^{-8}$

Table 4.1: Comparison of the final loss (masked MSE) achieved by the 2D U-net if trained to interpolate regularly downsampled input images with 50%, 25% and 12.5% of original data, respectively.

of the input data, performing zero-padding and finally coming back to the original domain through the inverse Fourier transform, obtaining an up-sampled version as output. In the 2D case we compared the U-net performances also with those of a BCI (Sec. 1.2), that is one of the most used model-based interpolation schemes, while in the 3D case we proposed a BLI (Sec. 1.2) as an alternative competitor.

4.3 2D U-net

4.3.1 Regularly downsampled input images

We trained, validated and tested the U-net on three datasets differing in the percentage of original data present within the input images, i.e. differing in the level of corruption. More precisely, input images had the 50% of original data in the first dataset, 25% of original data in the second one and 12.5% in the third dataset. During a first analysis, training, validation and test sets where composed by input images with the same percentage of missing data, depending on the dataset considered among the three available. In order to vary the data percentage when preparing the three datasets, we simply applied the deterministic binary mask $\mathbf{D}$ defined in (3.9) to the original images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{HR})}$, assigning in the definition of $\mathcal{L}_{\mathbf{x}}$ a downsampling factor d_x equal to 2, 4 and 8, thus obtaining regularly downsampled images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ with 50%, 25% and 12.5% of original data respectively.

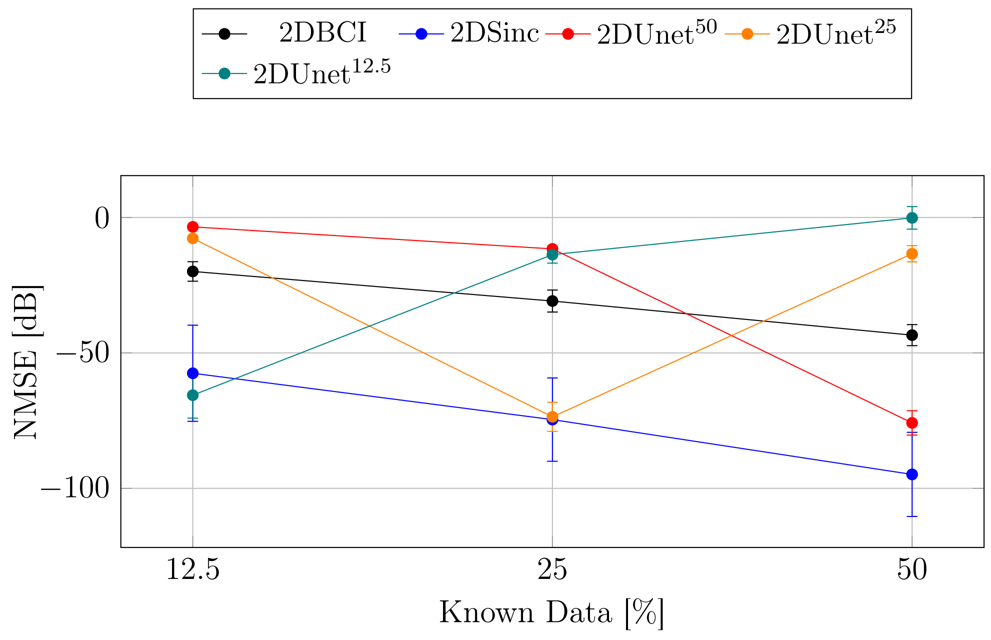
In Table 4.1 are reported the final training loss values, i.e the masked MSE defined in Sec. 2.3, achieved by the 2D U-net, when dealing with regularly downsampled input images having 50%, 25% and 12.5% of original data, respectively. By looking at the table, it can be noticed a worsening of the final training loss by increasing the percentage of missing data in the input images. This progressive worsening of the loss was expected. Our purpose was indeed that one of checking the performances of our model when dealing with progressively higher level of data corruption, in order to see its limits. This table is meaningful, because it puts in evidence that the 2D U-net still maintains pretty good performances even in the worst case analyzed by us, i.e when reconstructing input images with only 12.5% of original data.

After we trained the 2D U-net on the three datasets, we decided to check the U-net performances when tested on images with a different percentage of data with respect to that one used for training. More precisely, from the training campaign we ended up with three models: a 2DUnet^{50} model, which is the 2D U-net trained on images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ with 50% of original data; a 2DUnet^{25} model, which is the 2D U-net trained on images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ with 25% of original data; a 2DUnet$^{12.5}$ model, which is the 2D U-net trained on images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ with 12.5% of original data. What we have done was to test each of these three models on three test sets, composed by regularly downsampled input images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ having 12.5%, 25% and 50% of original data, respectively. We analyzed the performances in terms of average value and standard deviation of both NMSE and NCC, evaluating those metrics as a function of the percentage of known data in the test set. Results have been compared both with those of a BCI and a Fourier-based interpolation scheme. Metrics results are shown in Fig. 4.1. What can be noticed by looking at the NMSE behaviour, is that the best performances are achieved by the Fourier-based interpolation scheme, apart when tested on downsampled images with 12.5% of original data, where the 2DUnet$^{12.5}$ model shows to give the best results. It can be clearly observed that the U-net reaches good NMSE values, around -65 dB, when tested on regularly downsampled images with the same amount of data as in the training set, overcoming the performances provided by the BCI. However, the results get considerably worse for the U-net, if tested on images with a percentage of original data that differs from that one used in the training set. This evidences a limit of our proposed method, that is actually overfitting. As the U-net is trained to interpolate images that have been downsampled on a fixed regular grid, when it is tested on images with a downsampling scheme that has not be encountered during training, the predictions are poor. The U-net is thus not able to learn features that does not depend on the specific downsampling grid used. In the next section we will see how the use of random binary masks applied on the 2D images will help our models in overcoming this issue.

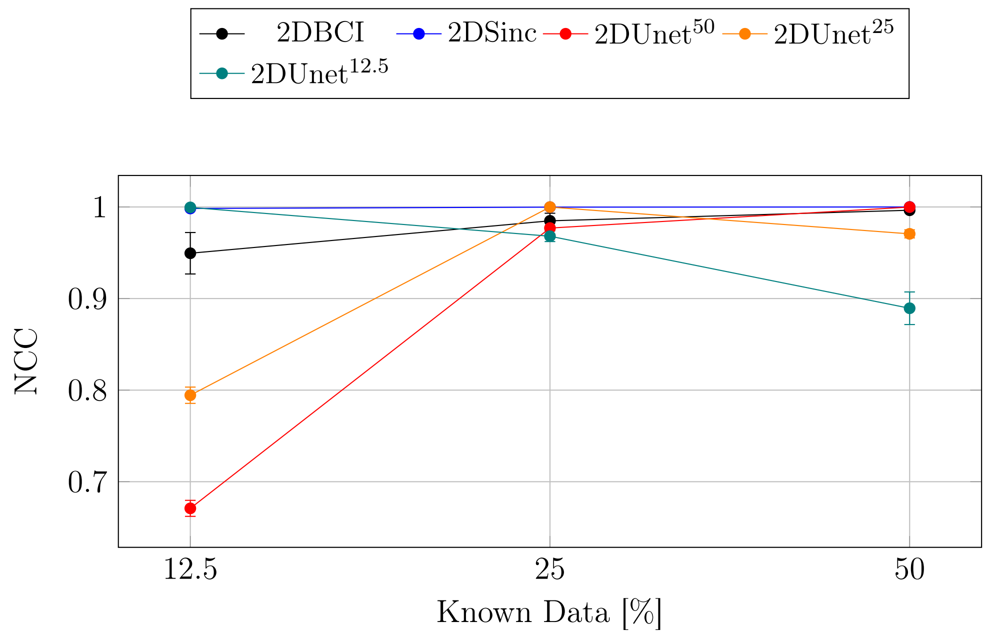
4.3.2 Irregularly sampled input images

In this case study, in order to vary the data percentage in the input images for the three datasets used to train the 2D U-net, we applied the 2D random binary mask $\mathbf{D}$ defined in (3.10) to the original images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{HR})}$, where we assigned to the probability p_x, in the definition of $\mathbf{b_x} \sim \mathcal{B}(N, p_x)$, values equal to 2, 4 and 8, to simulate irregularly sampled images $\mathbf{I}_{\mathrm{xf}}^{(\mathrm{LR})}$ with 50%, 25% and 12.5% of original data, respectively.

In Table 4.2 are reported the values of the final training loss, i.e the masked MSE defined in Sec. 2.3, achieved by the 2D U-net, compared as a function of the percentage of original data in the irregularly sampled images given as input to the model. It can be noticed that the values of the final training loss are worse than those achieved in the case of



(a)



(b)

Figure 4.1: Post-training metrics comparison varying the percentage of known data in the test set of regularly downsampled 2D space-frequency images. In (a) NMSE average shown along with its standard deviation and in (b) NCC average shown along with its standard deviation. On the x-axis the amount of known data in the test set, in the legend the amount of known data in the training set.

	50%	**25%**	**12.5%**
Train Loss	$4.5 \cdot 10^{-8}$	$9 \cdot 10^{-8}$	$2.8 \cdot 10^{-7}$

Table 4.2: Comparison of the final loss (masked MSE) achieved by the 2D U-net if trained to interpolate irregularly sampled input images with 50%, 25% and 12.5% of original data respectively.

regularly downsampled input images (see Table 4.1). This actually was expected, as in this case the U-net is trained to interpolate irregularly sampled images, which is actually a more complex scenario if compared to that one discussed in the previous section. Moreover, as already discussed in Sec. 3.3, the irregular grids through which input images have been corrupted are random, so the 2D U-net is forced to learn FRFs patterns independently on the specific grid used and, as can be imagined, this makes the task even more complicated to be pursued. Still, final training loss values are comparable to those achieved in the regular case.

In Fig. 4.2 are shown a target space-frequency image $\mathbf{I}_{\text{xf}}^{(\text{HR})}$, the relative irregularly sampled counterpart $\mathbf{I}_{\text{xf}}^{(\text{LR})}$ with 25% of original data, and the reconstructions provided both by our 2D U-net trained on irregularly sampled images with 25% of original data, and a 2D Fourier-based interpolation scheme. In Fig. 4.3, a FRF taken from the same space-frequency image and the relative reconstructions are shown. It can be easily seen that the 2D U-net provides visually better reconstructions than those of the Fourier-based competitor. Analyzing specifically Fig. 4.2, it is noticeable the weakness of the Fourier-based interpolation scheme in reconstructing missing FRF data in correspondence of the spatial edges. Fig. 4.3 reveals that FRF spikes are well reconstructed by our U-net, thanks to the contribution of $\mathbf{M_1}$ in the definition of the loss weight mask $\mathbf{M}$ (Sec. 2.3). On the other hand, the contribution of the edge mask $\mathbf{M_3}$ allows to provide a good reconstruction in correspondence of the spatial edges, overcoming the limits shown by the Fourier-based competitor, thus showing the ability to preserve the information strictly related to the boundary conditions imposed.

After the training on the three dataset was completed and results analyzed, we also decided to test the U-net performances in terms of average value and standard deviation of both the NMSE and the NCC, evaluating those metrics as a function of the percentage of known data in the test set. We thus built nine test sets composed by irregularly sampled input images having a specific percentage of original data, going from 10% to 90%. In order to do so, we have again applied the 2D random binary mask defined in (3.10) to the original images $\mathbf{I}_{\text{xf}}^{(\text{HR})}$, where the probability p_x, in the definition of $\mathbf{b_x} \sim \mathcal{B}(N, p_x)$, has been changed from 0.1 to 0.9. The tests have been performed for the 2D U-net previously trained on images having a percentage of original data equal to 50%, 25% and 12.5%,

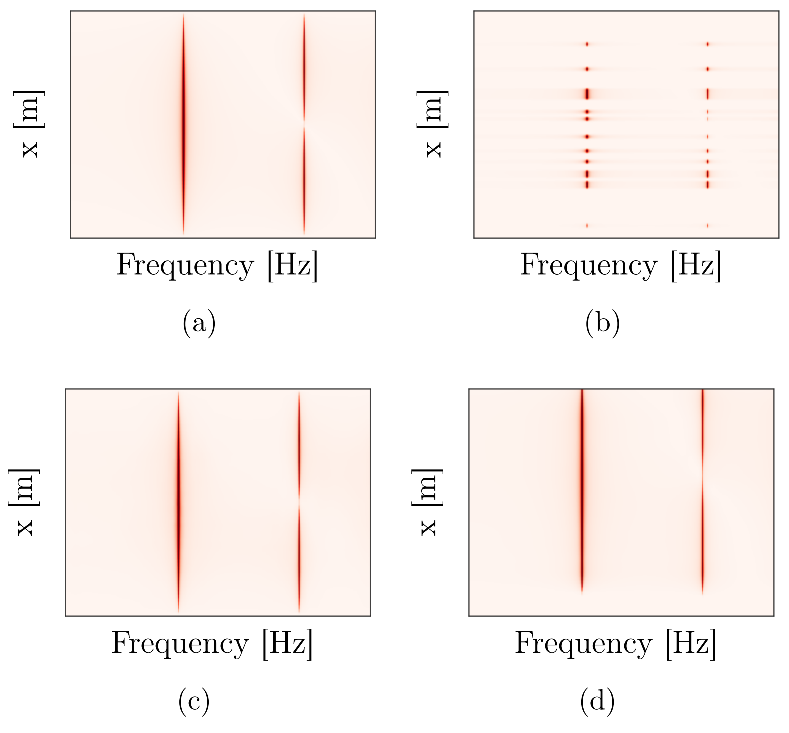


Figure 4.2: 2D space-frequency reconstructions comparison. In (a) an target space frequency image $\mathbf{I}_{\text{sf}}^{(\text{HR})}$; in (b) the relative irregularly sampled counterpart $\mathbf{I}_{\text{sf}}^{(\text{LR})}$ with 25% of original data, and the reconstructions provided by (c) the 2D U-net trained on images with 25% of original data and (d) a 2D sinc interpolation scheme.

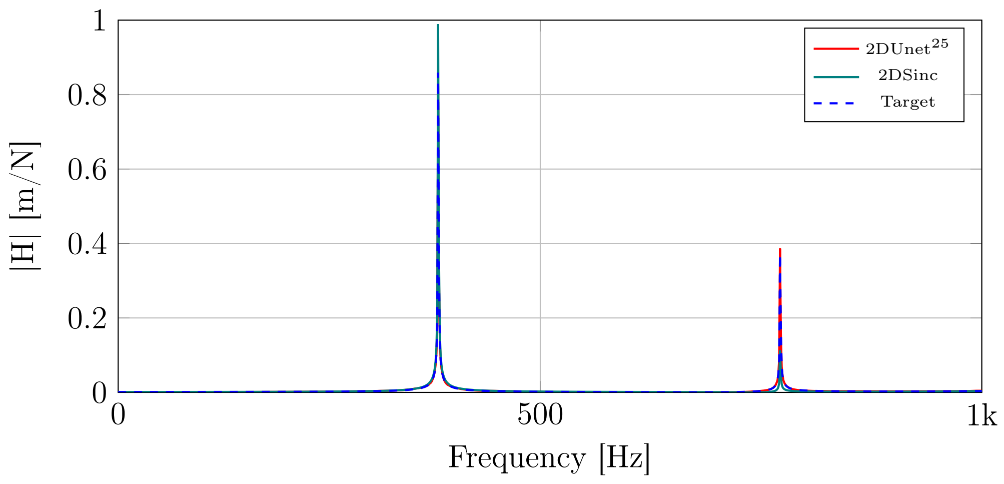


Figure 4.3: Example of reconstructed FRF signal from a synthetic space-frequency image. Target FRF in blue (dashed line), 2D U-net reconstruction (red line) and reconstruction provided by 2D sinc interpolation scheme (teal line).

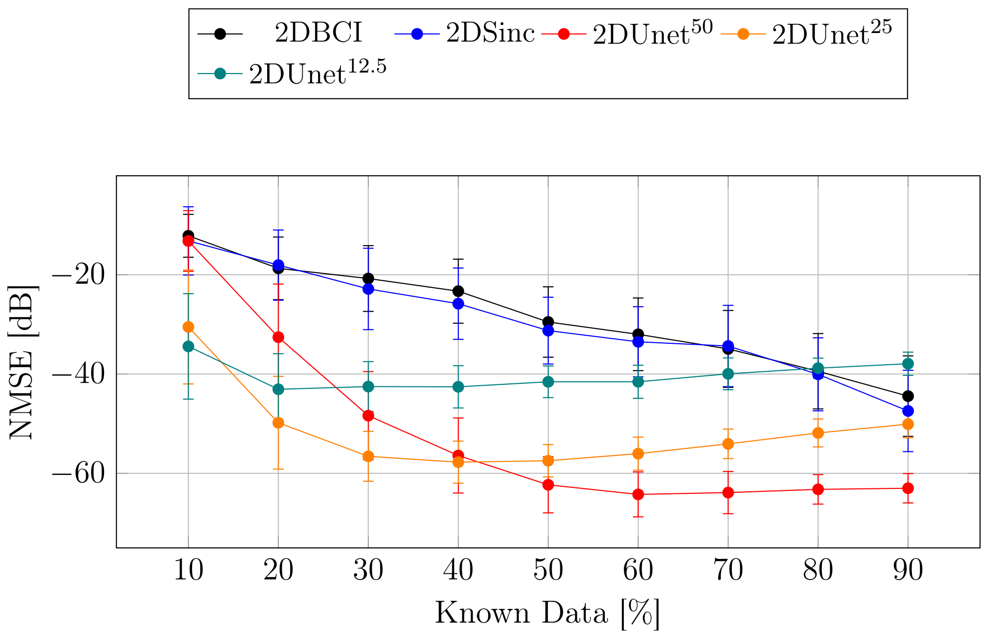
respectively, comparing again the performances with those of a BCI and a Fourier-based interpolation scheme. The resulting metrics are shown in Fig. 4.4. Looking specifically at the metrics in terms of NMSE, it can be observed a turnaround with respect to what has been seen in the previous section for regularly downsampled images. While both the BCI and the Fourier-based interpolation schemes present a pretty linear behaviour with respect to the percentage of original data in the test input images, the 2D U-net shows a plateau for percentages values larger than the one used when training the U-net. 2DUnet^{50} gives best performances for percentages near to 50%, 2DUnet^{25} gives best performances for percentages near to 25% and 2DUnet$^{12.5}$ gives best performances for percentages near to 12.5%. In any case, still when testing on input images having 80% of original data, all the three pre-trained models performs better than both the BCI and the Fourier-based interpolation scheme, independently on the percentage of original data present in the input images when training. Also by observing the NCC behaviour, it is evident that our models always overcome the competitors, a part for the case of the 2DUnet^{50} when tested on images with 10% of original data. Our 2D U-net provides a deeper understanding of the FRF patterns present in the data. From these results, we can conclude that both the model-based interpolation schemes proved to give the best performances if applied on regularly downsampled images, but their performances strongly drop when dealing with data affected by irregular sampling schemes, where, on the other hand, our model shows to be stronger, achieving reliable results.

4.4 3D U-net

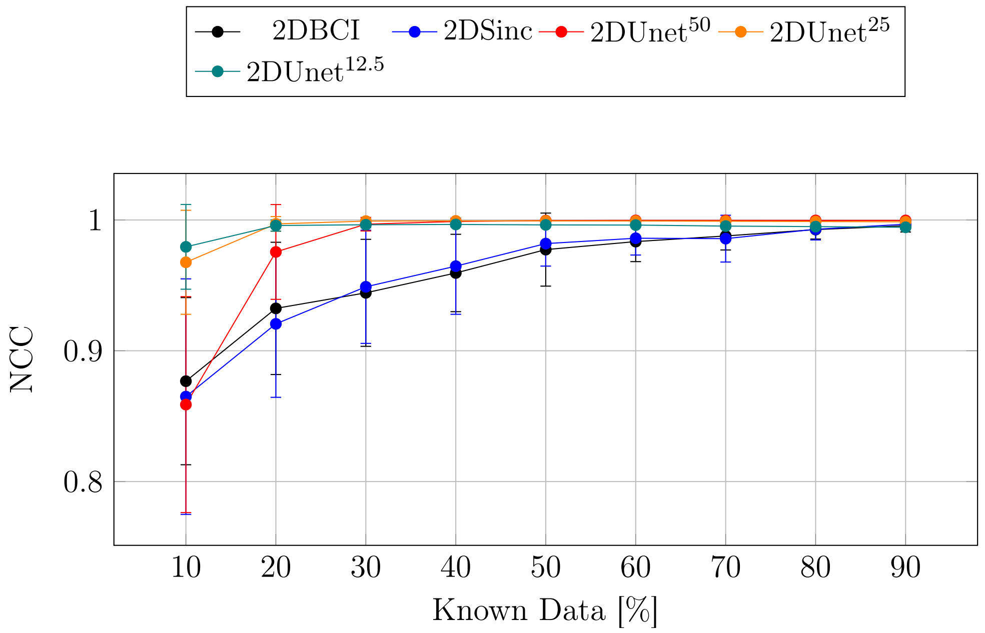
4.4.1 Regularly downsampled input tensors

As done in the 2D case, we trained, validated and tested the 3D U-net on three datasets, differing in the percentage of original data in the input tensors. Specifically, input tensors belonging to the first dataset had the 50% of original data, those belonging to the second dataset had the 25% of original data and finally, input tensors belonging to the third dataset had the 12.5% of original data. Given a specific dataset among the three defined, training, validation and test sets where composed by input tensors with the same percentage of missing data. In order to vary this percentage, we applied to the original tensors $\mathbf{H}^{(\mathrm{HR})}$ the 3D mask $\mathbf{D}$, obtained by extending the 2D deterministic binary mask $\mathbf{B}$ in (3.5) along the frequency axis, assigning in the definition of $\mathcal{L}_{\mathbf{x}}$ and $\mathcal{L}_{\mathbf{y}}$, pairs of values (2,1), (2,2) and (4,2) to the downsampling factors (d_x, d_y), with the aim to simulate regularly downsampled tensors $\mathbf{H}^{(\mathrm{LR})}$ with 50%, 25% and 12.5% of original data, respectively.

In Table 4.3 the values of the final training loss, i.e. the masked MSE defined in Sec. 2.3, are reported, compared as a function of the percentage



(a)



(b)

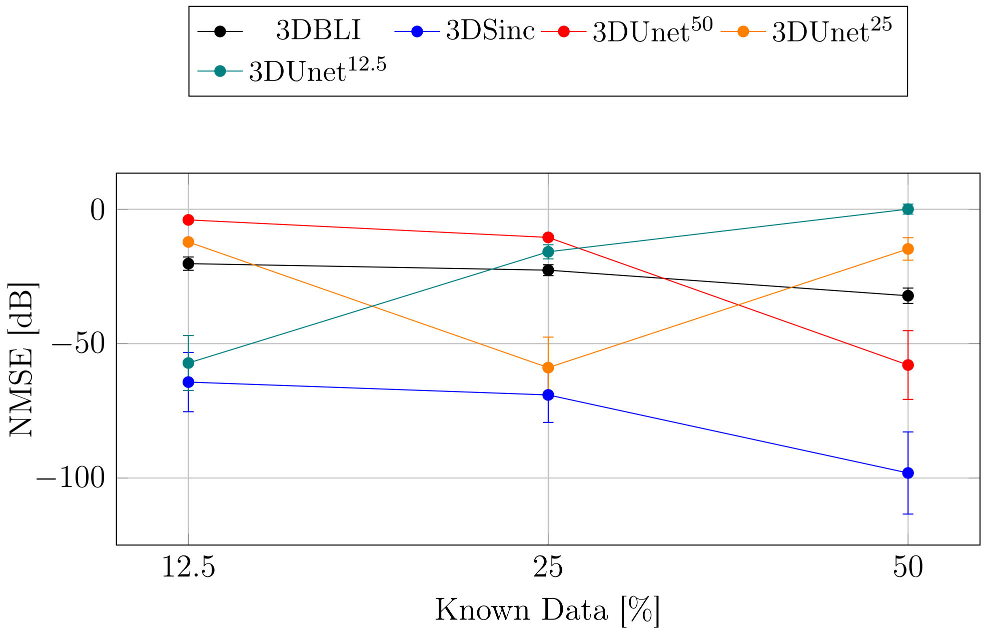
Figure 4.4: Post-training metrics comparison varying the percentage of known data in the test set of irregularly sampled 2D space-frequency images. In (a) NMSE average shown along with its standard deviation and in (b) NCC average shown along with its standard deviation. On the x-axis the amount of known data in the test set, in the legend the amount of known data in the training set.

	50%	**25%**	**12.5%**
Train Loss	$7.4 \cdot 10^{-7}$	$9.4 \cdot 10^{-7}$	$1 \cdot 10^{-6}$

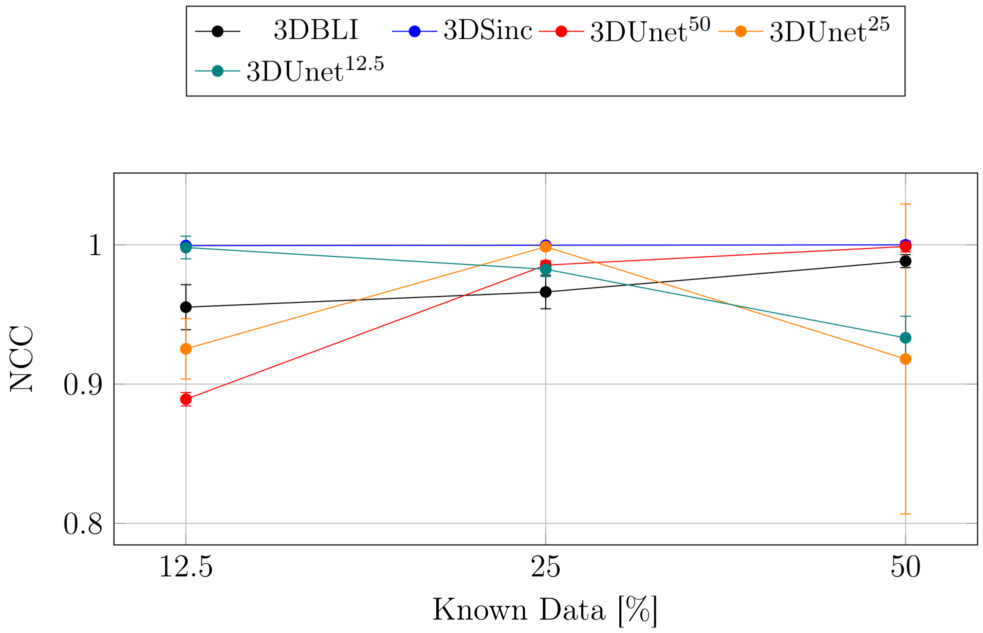
Table 4.3: Comparison of the final training loss (masked MSE) achieved by the 3D U-net if trained to interpolate regularly downsampled tensors having 50%, 25% and 12.5% of original data respectively.

of original data in input tensors. By looking at the table, it can be noticed a worsening of the final training loss by increasing the percentage of missing data in the input tensors. This progressive worsening of the loss was expected. As done in the 2D case, our purpose was to check the performances of our model when dealing with progressively higher level of data corruption, in order to see its limits. The 3D U-net actually maintains good performances even in the worst case analyzed, i.e when reconstructing input tensors with only 12.5% of available data.

After training the 3D U-net on the three datasets, we decided to test its performance on regularly downsampled tensors with a percentage of original data different from the one characterizing training tensors. More precisely, from the training campaign we ended up with three models: 3DUnet^{50} is the 3D U-net trained on tensors $\mathbf{H}^{(\mathrm{LR})}$ with 50% of original data; 3DUnet^{25} is the 3D U-net trained on tensors $\mathbf{H}^{(\mathrm{LR})}$ with 25% of original data; 3DUnet$^{12.5}$ is the 3D U-net trained on tensors $\mathbf{H}^{(\mathrm{LR})}$ with 12.5% of original data. What we have done was to test each of these three models on three test sets, composed by regularly downsampled input tensors $\mathbf{H}^{(\mathrm{LR})}$ having 12.5%, 25% and 50% of original data, respectively. We analysed the performances in terms of average value and standard deviation of both NMSE and NCC, evaluating those metrics as a function of the percentage of known data in the test set. Results have been compared both with those of a BLI and a Fourier-based interpolation scheme. Metrics results are shown in Fig. 4.5. As observed in the 2D case, when dealing with regularly downsampled input data, also in the case of 3D tensors the best results are achieved by the Fourier-based interpolation scheme, independently on the percentage of data present. Again it can be noticed that the U-net reaches good results, around -60 dB in terms of NMSE, when tested on tensors with the same amount of data as in the training set, overcoming the performances provided by the BLI. However, the results get considerably worse for the U-net, if tested on tensors with a percentage of original data that differs from that one used for training. This still evidences that also when dealing with 3D regularly downsampled tensors, the U-net actually overfits. As it is trained to interpolate tensors that have been downsampled through a fixed regular grid, our proposed model is not able to extract features that do not depend on the specific downsampling grid used, and, as a result, the predictions provided are really poor if testing is performed on



(a)



(b)

Figure 4.5: Post-training metrics comparison varying the percentage of known data in the test set of regularly downsampled 3D FRF tensors. In (a) NMSE average shown along with its standard deviation and in (b) NCC average shown along with its standard deviation. On the x-axis the amount of known data in the test set, in the legend the amount of known data in the training set.

	50%	**25%**	**12.5%**
Train Loss	$1.5 \cdot 10^{-6}$	$3.6 \cdot 10^{-6}$	$5.8 \cdot 10^{-6}$

Table 4.4: Comparison of the final training loss (masked MSE) achieved by the 3D U-net if trained to interpolate irregularly sampled tensors having 50%, 25% and 12.5% of original data respectively.

tensors with a regular downsampling scheme that has not be encountered during training. Also in this case, as will be demonstrated in the next section, the use of random binary masks applied on the 3D tensors will definitely help in overcoming this issue.

4.4.2 Irregularly sampled input tensors

In this case study, in order to vary the data percentage in the input tensors for the three datasets built to train the 3D U-net, we applied to the original tensors $\mathbf{H}^{(\mathrm{HR})}$ the 3D mask $\mathbf{D}$ obtained by extending the 2D random binary mask $\mathbf{B}$ in eq. 3.7 along the frequency axis, assigning in the definition of $\mathbf{b_x} \sim \mathcal{B}(N, p_x)$ and $\mathbf{b_y} \sim \mathcal{B}(M, p_y)$ values of (p_x, p_y) equal to (0.5,1), (0.5,0.5) and (0.5,0.25), with the aim to simulate irregularly sampled tensors $\mathbf{H}^{(\mathrm{LR})}$ with 50%, 25% and 12.5% of original data, respectively.

In Table 4.4 the values of the final training loss are reported, i.e the masked MSE defined in Sec. 2.3 achieved by the 3D U-net, evaluated as a function of the percentage of original data present in the input tensors, 50%, 25% and 12.5% respectively. As observed in the 2D case, it can be noticed that the values of final training loss are a worse of those achieved in the case of regularly downsampled input tensors (see Table 4.3). This actually was again expected, as in this case the U-net is trained to interpolate irregularly sampled tensors, which is actually an harder scenario to deal with if compared to that one of interpolating regularly downsampled tensors. Moreover, as done in the 2D analysis, the irregular grids through which input tensors have been corrupted are even random, so the 3D U-net is forced to learn FRF patterns independently on the specific grid used.

In Fig. 4.6 it can be seen a space-frequency slice taken from a target tensor $\mathbf{H}^{(\mathrm{HR})}$, a slice taken in correspondence of the same frequency bin belonging to the relative irregularly sampled counterpart $\mathbf{H}^{(\mathrm{LR})}$ with 25% of original data, and the reconstructions provided both by our 3D U-net trained on tensors with 25% of original data and a 3D Fourier-based interpolation scheme. In Fig. 4.7, a target FRF signal and the relative reconstructions are shown in detail. It can be easily seen that the reconstructions provided by the 3D U-net are visually better than those of the Fourier-based competitor. Analyzing specifically Fig. 4.6, it is noticeable that the Fourier-based interpolation scheme fails in retrieving

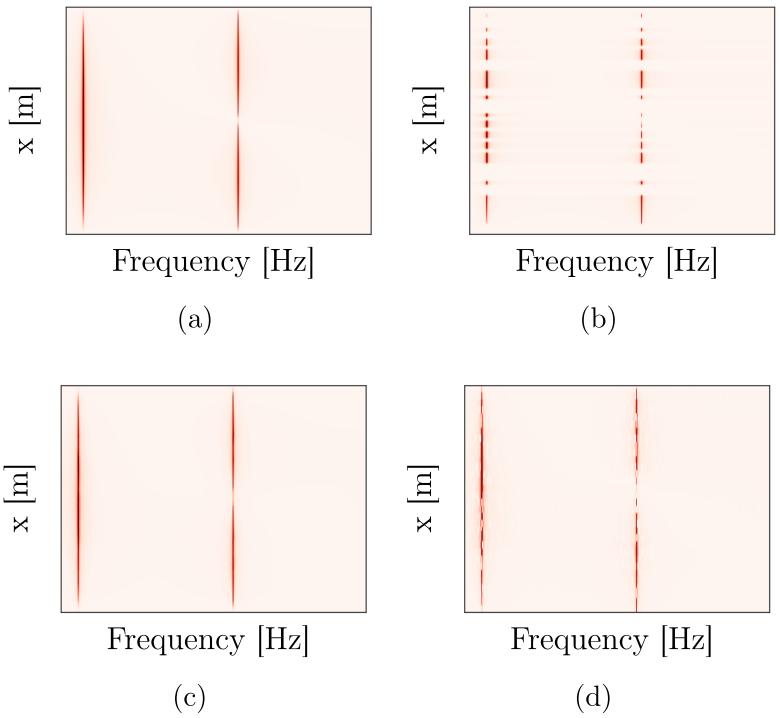


Figure 4.6: 3D tensor reconstructions comparison. In (a) a space-frequency slice from a target tensor $\mathbf{H}^{(\mathrm{HR})}$; in (b) the slice taken from the corresponding irregularly sampled tensor $\mathbf{H}^{(\mathrm{LR})}$ with 25% of original data, and the reconstructions provided by (c) our 3D U-net and (d) a 3D sinc interpolation scheme.

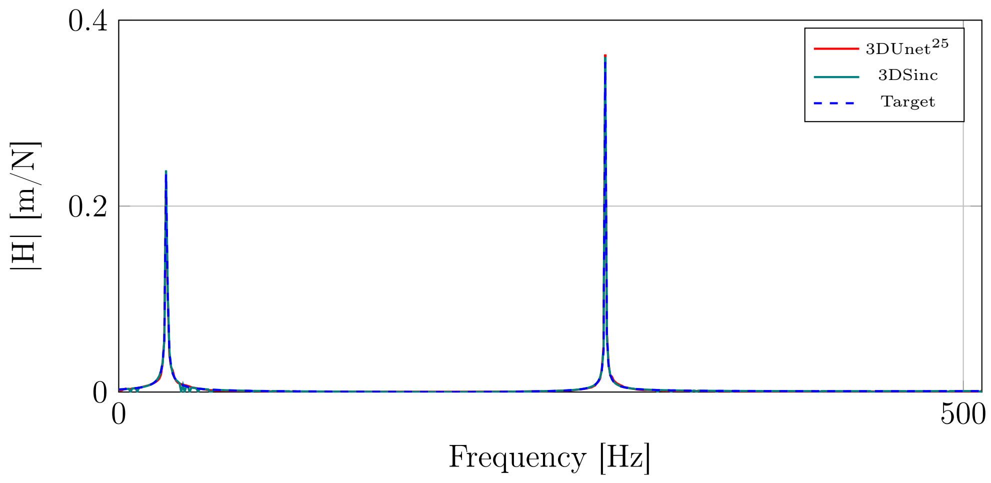
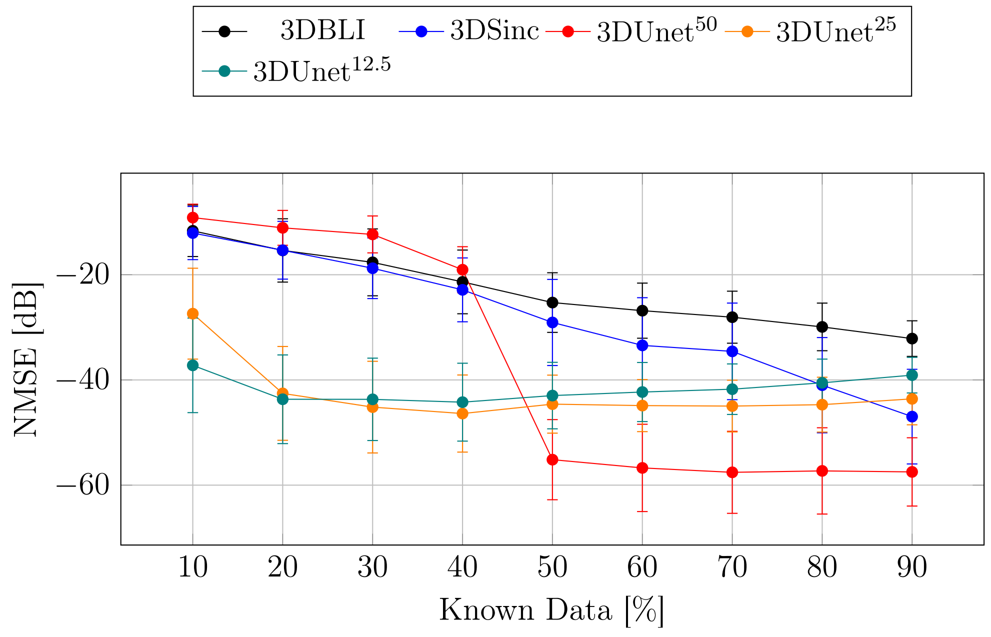


Figure 4.7: Example of reconstructed FRF signal from a synthetic tensor. Target FRF in blue (dashed line), 3D U-net reconstruction (red line) and reconstruction provided by 3D sinc interpolation scheme (teal line).

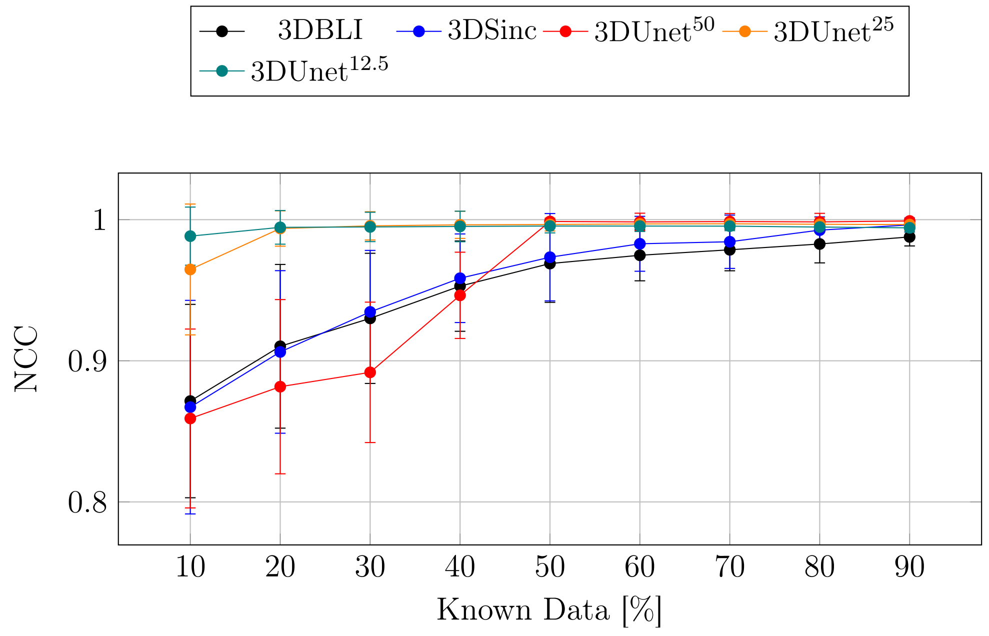
information in correspondence of the spatial edges, thus not being able to approximate the boundary conditions imposed on the original system. On the other hand, the 3D U-net reveals to be more reliable for this particular task, due to the introduction of $\mathbf{M_3}$ in the definition of the loss function mask $\mathbf{M}$, which imposes a prior in correspondence of the spatial edges. At the same time, both spikes and valleys are better reconstructed by the 3D U-net, and this is due to the combined usage of $\mathbf{M_1}$ and $\mathbf{M_2}$ in the definition of the loss function mask $\mathbf{M}$, giving prior weights to resonances and antiresonances, respectively.

This results, achieved by both our U-nets when dealing with irregularly sampled data, are particularly promising if contextualized in the field of vibration studies. Having proper reconstructions in correspondence of the spikes, that in our case represents the magnitude of the resonances associated to the plate under analysis, is of primarily importance, as studies like EMA relies on parameters like position, amplitude and width of the FRF spikes in order to find the proper modal parameters of the system. Having an algorithm able to reconstruct spikes properly positioned along the frequency range and with proper width and amplitude, means retrieving meaningful modal parameters when performing EMA, thus accessing the real dynamic properties of a system.

After the training on the three datasets was completed and results analyzed, we also decided to test the U-net performances, in terms of the average value and standard deviation of both the NMSE and the NCC, evaluating those metrics as a function of the percentage of known data in the test set. Nine test sets have thus been built, with input tensors $\mathbf{H}^{(\mathrm{LR})}$ having a percentage of original data going from 10% to 90%. In order to do so, we have again applied to the original tensors $\mathbf{H}^{(\mathrm{HR})}$ the 3D mask $\mathbf{D}$, obtained by extending the 2D random binary mask $\mathbf{B}$ in (3.7) along the frequency axis, where the probabilities p_x and p_y in the definition of $\mathbf{b_x} \sim \mathcal{B}(N, p_x)$ and $\mathbf{b_y} \sim \mathcal{B}(M, p_y)$, have been varied such that their product results in probability values going from 0.1 to 0.9. The tests have been performed for the 3D U-net previously trained on tensors having a percentage of original data equal to 50%, 25% and 12.5%, respectively, comparing again the performances with those of a BLI and a 3D Fourier-based interpolation scheme. Metrics results are shown in Fig. 4.8. From these plots, it can be understood that the scenario is radically changed from the one observed when dealing with regularly downsampled tensors. The Fourier-based interpolation scheme indeed, that in the regular case gave the best performances, as shown in Fig. 4.5, turned out providing really poor reconstructions if tested on irregularly sampled tensors. By looking at the NMSE behaviour, both the BLI and the Fourier-based baselines are characterized by a linear trend, while in the case of the 3D U-net we can observe that 3DUnet$^{12.5}$ and 3DUnet^{25} are particularly good in correspondence of the percentage of original data used for training, i.e. 12.5% and 25% respectively, outperforming both the BLI and the Fourier-based interpolation scheme until the percentage



(a)



(b)

Figure 4.8: Post-training metrics comparison varying the percentage of known data in the test set of irregularly sampled 3D FRF tensors. In (a) NMSE average shown along with its standard deviation and in (b) NCC average shown along with its standard deviation. On the x-axis the amount of known data in the test set, in the legend the amount of known data in the training set.

of original data in the test tensors is lower than 80%. On the other hand, the 3DUnet^{50} gives the best performances for percentages higher that 50%, while getting considerably worse for percentages lower than 50%. This means that in this latter case, the U-net is able to interpolate irregular sampling grids with high accuracy until the amount of known data is greater or equal than the one used during the network training. The same observations can be done when analysing the NCC behaviour, where again poor performances are provided by the 3DUnet^{50} if tested on test sets characterized by percentages of data going from 10% up to 50%, while giving the best performances for percentages higher or equal to 50%.

By taking all these considerations, it can be said that, when dealing with tensors having a percentage of original data ranging from 20% to 80%, the best choice would be to use 3DUnet$^{12.5}$ or 3DUnet^{25} as preferred interpolation methods, given that they provide performances which are pretty stable with respect to the level of input data corruption.

4.5 2D U-net applied on real FRF data

Finally, in order to evaluate the performances of the proposed architectures on a real scenario, we tested our 2D U-net on FRFs acquired from a real wooden rectangular plate. The system response to hammer impacts was measured on 32 aligned points along the plate width. More precisely, the test has been conducted following the roving hammer approach, i.e. keeping fixed the excitation point and moving an accelerometer to acquire response signals over 32 points along the plate width. The plate time responses have been acquired for a duration of 2 seconds and with a 48 kHz sampling frequency. The FRFs have thus been obtained by computing the frequency conterparts of the acquired signals and deconvolving them by the frequency counterpart of the excitation signal. Each FRF has been thus defined in the range $[0.5\text{Hz}, 512\text{Hz}]$ with a frequency step of 0.5 Hz. The 32 FRFs obtained in correspondence of the 32 spatial points have thus been stacked, obtaining a regularly sampled space-frequency image $\mathbf{I}_{\text{xf}}^{(\text{HR})} \in \mathcal{R}^{32\times1024}$ as a result of the vibrometric study.

The low resolution counterpart $\mathbf{I}_{\text{xf}}^{(\text{LR})}$ has been obtained with a 2D random binary mask $\mathbf{D}$ with 25 % of available data. We chose this percentage, as in our opinion it could resemble a realistic scenario. Indeed, a percentage of original data equal to the 50% could be a too optimistic case to deal with, while, on the other hand, an image with only the 12.5% of original data represents a particularly bad scenario, that in real application should not be encountered.

In Fig. 4.9 it can be seen the target space-frequency image $\mathbf{I}_{\text{xf}}^{(\text{HR})}$, the relative irregularly sampled counterpart $\mathbf{I}_{\text{xf}}^{(\text{LR})}$ with 25% of original data, and the reconstructions provided both by our 2D U-net trained on images with 25% of original data and a 2D Fourier-based interpolation

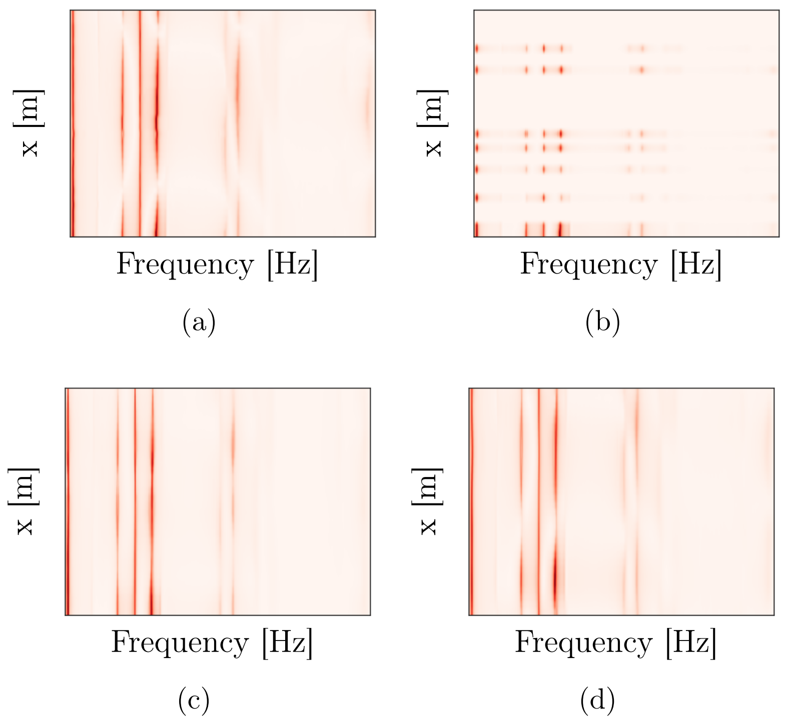


Figure 4.9: Reconstructions comparison of a 2D space-frequency image obtained by real measurements. In (a) a target image $\mathbf{I}_{\text{xf}}^{(\text{HR})}$; in (b) the irregularly sampled counterpart $\mathbf{I}_{\text{xf}}^{(\text{LR})}$ with 25% of original data, and the reconstructions provided by (c) our 2D U-net and (d) a 2D sinc interpolation scheme.

scheme. In Fig. 4.10, a target FRF signal and the relative reconstructions are shown in detail. As it can be observed in the plots, both techniques

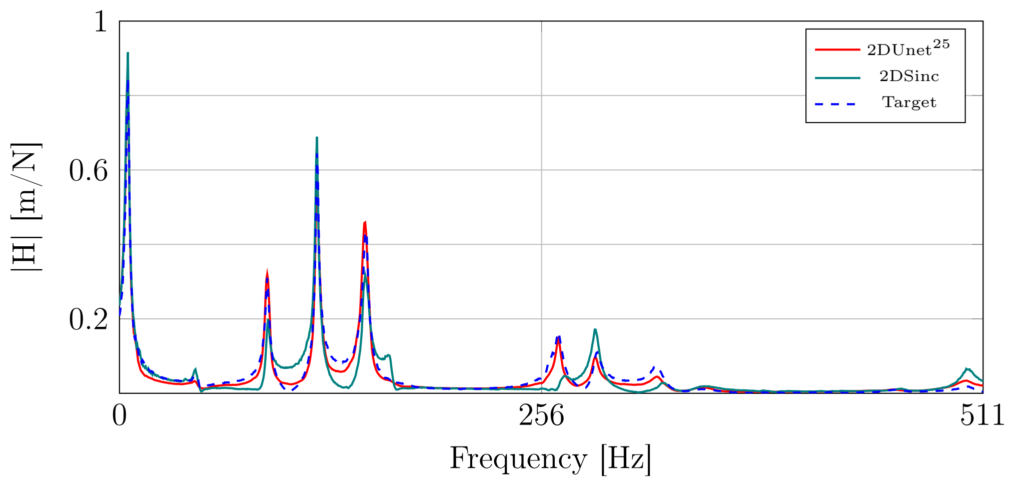


Figure 4.10: Example of reconstructed signal from real acquisitions on a rectangular wooden plate. Target FRF in blue (dashed line), 2D U-net reconstruction (red line) and BCI reconstruction (teal line).

correctly recover the position of the FRF peaks, but the Fourier-based interpolation scheme fails in generating the desired amplitude values. This can be an issue in curve-fitting techniques for EMA, where, as already discussed, the peaks amplitudes are used as initial guesses for the optimization problem in order to retrieve the system modal parameters. On the other hand, the U-net proves to be able in generating good amplitude values.

The metrics computed on the reconstructed image show that the 2D U-net still provides the best performance, achieving NMSE $= -21\,\mathrm{dB}$ and NCC $= 0.96$ compared to NMSE $= -11.3\,\mathrm{dB}$ and NCC $= 0.87$ obtained by the Fourier-based competitor.

5 Conclusions and Future Works

In this thesis we proposed a method for the interpolation of both regularly down-sampled and irregularly sampled FRF data, based on Convolutional Autoencoders. We found a competitive solution to problems that are typical when acquiring vibration measurements, also in the field of Musical Acoustics and particularly in the study of violins plates. We decided to implement a specific type of CA, i.e. the U-net, taking advantage of the competitive results achieved in the field of inverse problems by this architecture. We started by defining a FEM model of a thin plate, acquiring FRF data in terms of receptance through the numerical methods implemented by the COMSOL Multiphysics® software, building a wide dataset of plates frequency responses. Data have been organized both as 3D tensors, containing FRFs signals evaluated on a dense spatial grid over the plate surface, and in 2D space-frequency slices. We corrupted those data structures both by means of regular and randomic downsampling masks, in order to prepare the LR inputs and train in parallel two U-net architectures, on 3D and 2D data respectively, to reconstruct the corresponding regularly sampled HR data.

Numerical results obtained on synthetic FRFs showed that both the U-net architectures are able to interpolate 2D and 3D regularly downsampled data unseen during training, with competitive performances only if the amount of information contained in the LR test samples is comparable to that one used for training. On the other hand, our models showed to efficiently interpolate 2D and 3D irregularly sampled data unseen during training, until the amount of information is greater or equal than the one contained in the training LR samples, outperforming the classical model-based interpolation methods used as competitors in any test case analyzed. This evidences a weakness of the latter algorithms, as their performances are really good when dealing with regularly down-sampled data, while dropping considerably if tested on irregular grids. On the other hand, our models become the best choices when dealing with particularly corrupted and irregularly sampled FRF data, which is actually a pretty interesting result, as in real applications it is for sure more common to have missing data disposed on irregular grids instead of regular downsampling scenarios.

We also tested the 2D U-net on really measured vibration data unseen during training, experimentally acquired from a real thin wooden plate. Results showed that our method is able to achieve a reconstruction error that is 10 dB less than the baseline technique and a good estimation of the peak amplitudes, which is desirable in EMA. This gave us the opportunity to observe that the proposed architectures are able to interpolate real measured data, even if training and validation have been performed on synthetic FRFs presenting inevitably different dynamics with respect to those of real data. Particularly interesting is also the fact that our models shown to be blind with respect to the materials of the system considered and thus to their elastic properties. Indeed, good reconstructions have been achieved on a real orthotropic plate, even if our models have been trained and validated on FRFs data relative to synthetic isotropic plates. The efficiency of a DL method is strictly related to its ability of generalization, and these test proved that our models are pretty much on that direction.

All the results obtained are benefits of particular design choices taken during our work. The U-net itself have confirmed in this application its efficiency as a tool for solving inverse problems, and its ability to learn meaningful features from data thanks to an encoder-decoder architecture and to the presence of skip connections. At the same time, the definition of a weighted MSE as custom loss functions for training, was determinant in order for both the 2D and the 3D models to provide good reconstructions of spikes and valleys, also allowing to achieve an intrinsic knowledge of the boundary conditions imposed. Finally, the introduction of both Reflection padding and Batch Normalization layers, strongly helped the models in overcoming the problem of dealing with sparse data, preventing them to get stuck in local minima.

In future we aim to test our methodology on structures with arbitrary geometries, not simply limiting the study on rectangular thin plates, and to extend the interpolation procedure to the phase of the FRFs, searching suitable prior masks to be weight the loss function in order for the models to better fit phase data.

Bibliography

[1] C. M. Hutchins, "The acoustics of violin plates," *Scientific American*, vol. 245, no. 4, pp. 170–187, 1981.

[2] G. Squicciarini, R. Corradi, and e. al, "Modal analysis of a grand piano soundboard," in *International Conference on Sound and Vibration Engineering (ISMA-USD)*, 2010.

[3] J. Grosel, W. Sawicki, and W. Pakos, "Application of classical and operational modal analysis for examination of engineering structures," *Procedia Engineering*, vol. 91, 2014.

[4] B. Schwarz and M. H. Richardson, "Experimental modal analysis," in *Proceedings of CSI Reliability Week*, 1999.

[5] P. Avitabile, "Experimental modal analysis - a simple non-mathematical presentation," *Sound and Vibration*, vol. 35, pp. 20–31, 2001.

[6] A. Cunha and E. Caetano, "Experimental modal analysis of civil engineering structures," *Sound and Vibration*, vol. 40, 2006.

[7] G. Chardon, A. Leblanc, and L. Daudet, "Plate impulse response spatial interpolation with sub-nyquist sampling," *Journal of Sound and Vibration*, vol. 330, pp. 5678–5689, 2011.

[8] B. Schwarz, S. Richardson, and M. Richardson, "Curve fitting analytical mode shapes to experimental data," in *Topics in Modal Analysis & Testing, Volume 10* (M. Mains, ed.), pp. 45–59, Springer International Publishing, 2016.

[9] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[10] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, 2017.

[11] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883, 2016.

[12] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1132–1140, 2017.

[13] M. S. M. Sajjadi, B. Schölkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4501–4510, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[15] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, pp. 341–349, 2012.

[16] S. Mandelli, V. Lipari, P. Bestagini, and S. Tubaro, "Interpolation and denoising of seismic data using convolutional neural networks," *arXiv preprint arXiv:1901.07927*, 2019.

[17] K. Fukami, K. Fukagata, and K. Taira, "Super-resolution reconstruction of turbulent flows with machine learning," *Journal of Fluid Mechanics*, vol. 870, pp. 106–120, 2019.

[18] V. Kuleshov, S. Z. Enam, and S. Ermon, "Audio super-resolution using neural nets," in *Workshop of International Conference on Learning Representation*, 2017.

[19] M. S. Hossain, Z. C. Ong, Z. I., S. N., and S. Y. K., "Artificial neural networks for vibration based inverse parametric identifications: A review," *Applied Soft Computing*, vol. 52, pp. 203 – 219, 2017.

[20] C. Campagnoli, M. Pezzoli, F. Antonacci, and A. Sarti, "Vibrational modal shape interpolation through convolutional auto encoder," in *48th International Congress and Exposition on Noise Control Engineering*, I-INCE, 2020.

[21] D. E. Tsokaktsidis, T. von Wysocki, F. Gauterin, and S. Marburg, "Artificial Neural Network predicts noise transfer as a function of excitation and geometry," in *Proceedings of the 23rd International Congress on Acoustics: integrating 4th EAA Euroregio*, (Aachen, Germany), Sep 2019.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[23] G. Bissinger, "Structural acoustics of good and bad violins," *The Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1764–1773, 2008.

[24] C. J. Lomte, "Vibration analysis of anisotropic plates, special case: Violin," 2013.

[25] C. Gough, "Violin plate modes," *The Journal of the Acoustical Society of America*, vol. 137, no. 1, pp. 139–153, 2015.

[26] C. M. Hutchins, "A history of violin research," *The Journal of the Acoustical Society of America*, vol. 73, no. 5, pp. 1421–1440, 1983.

[27] T. Corridoni, M. D'Anna, and H. Fuchs, "Damped mechanical oscillator: Experiment and detailed energy analysis," *The Physics Teacher*, vol. 52, no. 2, pp. 88–90, 2014.

[28] O. Bauchau and J. Craig, *Kirchhoff plate theory*, vol. 163, pp. 819–914. 01 2009.

[29] N. H. Fletcher and T. D. Rossing, *The physics of musical instruments*. Springer Science & Business Media, 2012.

[30] F. Cheli and G. Diana, *Advanced dynamics of mechanical systems*. Springer, 2015.

[31] K. D. Marshall, "Modal analysis of a violin," *The Journal of the Acoustical Society of America*, vol. 77, no. 2, pp. 695–709, 1985.

[32] T. Duerinck, M. Kersemans, E. Skrodzka, M. Leman, G. Verberkmoes, and W. V. Paepegem, "Experimental modal analysis of violins made from composites," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, p. 535, 2018.

[33] Z.-F. Fu and J. He, *Modal analysis*. Elsevier, 2001.

[34] Y. Lu, *Comparison of finite element method and modal analysis of violin top plate*. PhD thesis, Citeseer, 2013.

[35] G. Cariolaro, "La teoria unificata dei segnali-nuova edizione," 2001.

[36] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang, "Super-resolution image reconstruction: a technical overview," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, 2003.

[37] A. W. M. van Eekeren, K. Schutte, and L. J. van Vliet, “Multiframe super-resolution reconstruction of small moving objects,” *Trans. Img. Proc.*, vol. 19, p. 2901–2912, Nov. 2010.

[38] P. Thévenaz, T. Blu, and M. Unser, “Image interpolation and resampling,” *Handbook of medical imaging, processing and analysis*, vol. 1, no. 1, pp. 393–420, 2000.

[39] X. Li and M. T. Orchard, “New edge-directed interpolation,” *IEEE transactions on image processing*, vol. 10, no. 10, pp. 1521–1527, 2001.

[40] R. Keys, “Cubic convolution interpolation for digital image processing,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[41] K. Jensen and D. Anastassiou, “Subpixel edge localization and the interpolation of still images,” *IEEE transactions on Image Processing*, vol. 4, no. 3, pp. 285–295, 1995.

[42] J. Allebach and P. W. Wong, “Edge-directed interpolation,” in *Proceedings of 3rd IEEE International Conference on Image Processing*, vol. 3, pp. 707–710, IEEE, 1996.

[43] D. D. Muresan and T. W. Parks, “Prediction of image detail,” in *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, vol. 2, pp. 323–326, IEEE, 2000.

[44] W. K. Carey, D. B. Chuang, and S. S. Hemami, “Regularity-preserving image interpolation,” *IEEE transactions on image processing*, vol. 8, no. 9, pp. 1293–1297, 1999.

[45] H. He and W.-C. Siu, “Single image super-resolution using gaussian process regression,” in *CVPR 2011*, pp. 449–456, IEEE, 2011.

[46] K. Zhang, X. Gao, D. Tao, and X. Li, “Single image super-resolution with non-local means and steering kernel regression,” *IEEE Transactions on Image Processing*, vol. 21, no. 11, pp. 4544–4556, 2012.

[47] W. T. Freeman, T. R. Jones, and E. C. Pasztor, “Example-based super-resolution,” *IEEE Computer graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[48] H. Chang, D.-Y. Yeung, and Y. Xiong, “Super-resolution through neighbor embedding,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, pp. I–I, IEEE, 2004.

[49] M. Aharon, M. Elad, and A. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE*

Transactions on signal processing, vol. 54, no. 11, pp. 4311–4322, 2006.

[50] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2008.

[51] S. Wang, L. Zhang, Y. Liang, and Q. Pan, "Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2216–2223, IEEE, 2012.

[52] L. He, H. Qi, and R. Zaretzki, "Beta process joint dictionary learning for coupled feature spaces with application to single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 345–352, 2013.

[53] T. Peleg and M. Elad, "A statistical prediction model based on sparse representations for single image super-resolution," *IEEE transactions on image processing*, vol. 23, no. 6, pp. 2569–2582, 2014.

[54] Y. Zhu, Y. Zhang, and A. L. Yuille, "Single image super-resolution using deformable patches," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2917–2924, 2014.

[55] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3791–3799, 2015.

[56] K. Zhang, D. Tao, X. Gao, X. Li, and J. Li, "Coarse-to-fine learning for single-image super-resolution," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1109–1122, 2016.

[57] J. Yu, X. Gao, D. Tao, X. Li, and K. Zhang, "A unified learning framework for single image super-resolution," *IEEE Transactions on Neural networks and Learning systems*, vol. 25, no. 4, pp. 780–792, 2013.

[58] C. Deng, J. Xu, K. Zhang, D. Tao, X. Gao, and X. Li, "Similarity constraints-based structured output regression machine: An approach to image super-resolution," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 12, pp. 2472–2485, 2015.

[59] W. Yang, Y. Tian, F. Zhou, Q. Liao, H. Chen, and C. Zheng, "Consistent coding scheme for single-image super-resolution via independent dictionaries," *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 313–325, 2016.

[60] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[61] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1646–1654, 2016.

[62] X. Mao, C. Shen, and Y.-B. Yang, “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections,” in *Advances in neural information processing systems*, pp. 2802–2810, 2016.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[64] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*, pp. 177–186, Springer, 2010.

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

[66] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[67] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher, *Computational intelligence: a methodological introduction*. Springer, 2016.

[68] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[69] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[70] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[71] F. Siddique, S. Sakib, and M. A. B. Siddique, “Recognition of handwritten digit using convolutional neural network in python with tensorflow and comparison of performance for various hidden layers,” in *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*, pp. 541–546, IEEE, 2019.

[72] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[73] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

[74] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[76] G. Liu, K. J. Shih, T.-C. Wang, F. A. Reda, K. Sapra, Z. Yu, A. Tao, and B. Catanzaro, "Partial convolution based padding," *arXiv preprint arXiv:1811.11718*, 2018.

[77] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[78] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[79] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[80] C. M. Bishop *et al.*, *Neural networks for pattern recognition.* Oxford university press, 1995.

[81] C. Sammut and G. I. Webb, eds., *Mean Squared Error*, pp. 653–653. Boston, MA: Springer US, 2010.

[82] N. Ponomarenko, S. Krivenko, K. Egiazarian, V. Lukin, and J. Astola, "Weighted mean square error for estimation of visual quality of image denoising methods," pp. 5–p, 2010.