Machine learning based simulation of realistic signals for an enhanced automatic diagnostic in non-destructive testing applications

Gerardo Emanuel Granados

université PARIS-SACLAY

Machine learning based simulation of realistic signals for an enhanced automatic diagnostic in non-destructive testing applications

Simulation réaliste basée sur des techniques d'apprentissage pour l'amélioration du diagnostic en contrôle non destructif

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°579 Sciences Mécaniques et Énergétiques, Matériaux et Géosciences (SMEMAG).
Spécialité de doctorat: Mécanique des solides.
Graduate School : Sciences de l'ingénierie et des systèmes.
Référent : CentraleSupélec.

Thèse préparée dans les unités de recherche **Laboratoire d'Intégration des Systèmes et des Technologies (CEA List)** et **LMPS - Laboratoire de Mécanique Paris-Saclay (Université Paris-Saclay, CentraleSupélec, ENS Paris-Saclay, CNRS)**, sous la direction de **Didier CLOUTEAU**, professeur à l'Université de Paris-Saclay, le co-encadrement de **Roberto MIORELLI**, ingénieur chercheur au CEA, et le co-encadrement de **Filippo GATTI**, maître de conférences à l'Université de Paris-Saclay.

Thèse soutenue à Paris-Saclay, le 14 novembre 2023, par

Gerardo Emanuel GRANADOS

Composition du jury

Membres du jury avec voix délibérative

Céline HUDELOT Professeure, Université Paris-Saclay	Présidente
Etienne DECENCIÈRE Directeur de recherche, École des Mines de Paris - Université PSL	Rapporteur & Examinateur
Stefano MARIANI Professeur, Politecnico di Milano	Rapporteur & Examinateur
Marie-Hélène AUMEUNIER Ingénieure de recherche, HDR CEA	Examinatrice
Bertrand IOOSS Ingénieur de recherche, HDR EDF R&D	Examinateur
Julie DIGNE Directrice de recherche, Université Claude Bernard Lyon I	Examinatrice

THESE DE DOCTORAT
NNT : 2023UPAST143

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Titre: Simulation réaliste basée sur des techniques d'apprentissage pour l'amélioration du diagnostic en contrôle non destructif.
Mots clés: simulation réaliste, techniques d'apprentissage, contrôle non destructif, diagnostic.

Résumé: Le développement d'outils de diagnostic automatique est un sujet de recherche très actif dans le domaine du contrôle non destructif, car il s'inscrit dans la stratégie de modernisation et de gestion améliorée des lignes de production au niveau européen. Ces outils visent à fournir à une chaîne de contrôle de plus haut niveau une évaluation qualitative ou quantitative de l'état du matériau inspecté (état sain, endommagé, dimensionnement, criticité de l'anomalie). L'institut CEA LIST est reconnu internationalement comme un acteur majeur de recherche dans le domaine du contrôle. Il développe la plateforme CIVA, qui est reconnue comme l'un des principaux logiciels de simulation multiphysique du domaine. Une modélisation fiable et précise des phénomènes physiques mis en jeu dans la mesure non destructive est un atout important dans une démarche de caractérisation des indications contenues dans le signal expérimental. Cependant, elle ne tient pas compte des perturbations et de la variabilité des entrées caractéristiques des expériences de mesure, c'est pourquoi on peut, par exemple, facilement distinguer un signal simulé « parfait » d'une acquisition expérimentale. Cette thèse porte sur le développement d'une solution permettant de réduire l'écart entre signaux simulés et expérimentaux, en augmentant les données de la simulation avec une contribution supplémentaire. Celle-ci peut être qualifiée comme « bruit » et représente tout ce qui est différent du signal physique déterministe régi par le jeu d'équations physiques correspondant à la mesure étudiée (ultrasons, électromagnétisme par example). La stratégie pour prendre en compte cette contribution consiste à appliquer des méthodes d'apprentissage à un jeu de données expérimental représentatif ou à entraîner un réseau de neurones à dissocier dans des acquisitions réelles le contenu (comme les signatures des défauts) du style (ce qui n'est pas simulé). Par la suite, cette simulation augmentée est utilisée dans des processus d'analyse de sensibilité, de gestion des incertitudes et de diagnostic automatique développés au CEA LIST. Elle permettra d'obtenir une meilleure adéquation entre la simulation et l'expérience, ainsi que la prise en compte de potentielles dérives cas-dépendantes dues à un environnement particulier.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Title: Machine learning based simulation of realistic signals for an enhanced automatic diagnostic in non-destructive testing applications.

Keywords: realistic simulation, machine learning, non-destructive testing, diagnostic.

Abstract: Model-based solutions for automatic diagnostics in the field of non-destructive testing are currently a topic of great interest in both academic and industial communities. Their ultimate objective is to provide a qualitative or quantitative evaluation of the inspected material state (sound, flawed, flawed with anomaly dimensions or criticality) in an industrial context like a production line. Such tools, providing inputs for real-time process control, contribute to the general trend in Europe that aims at modernizing industry and services. The CEA LIST Institute is an internationally recognized research institution in non-destructive testing and evaluation (NDT&E). It develops the CIVA software, which offers multi-physics models and is considered a leading product for simulation for NDT&E applications. Accurate models able to reproduce experimental signals prove very helpful in an inversion process aiming at classifying or characterizing flaws. However, as they do not account for disturbances and parameter variability occurring during an experimental acquisition, simulated signals inherently look "perfect" and are, for instance, easily distinguishable from experimental data. This PhD subject aims to improve the match between simulation and experimental data by augmenting the simulation with another contribution generally referred to as "noise". The strategy proposed to obtain such noise contribution is to apply machine-learning techniques to a set of representative experimental data. Alternatively, a deep learning model can be trained to analyze "real" data and distinguish between contents (flaw signals) and style (the rest, which physical models do not simulate). Afterwards, the augmented simulation tool will be able to reproduce closely experimental data, account for specific discrepancies due to a particular environment and reproduce the variability observed experimentally. It will thus enhance the performance of model-based tools developed at CEA LIST for sensitivity analysis, management of uncertainty and diagnostic.

Contents

Nomenclature

AdaIN	Adaptive Instance Normalization
AE	AutoEncoder
cAE	Conditional AutoEncoder
cGAAE	Class-conditioned Generative Adversarial AutoEncoder
CNN	Convolutional Neural Networks
cU-Net	Conditional U-Net
DA	Domain Adaptation
DL	Deep Learning
DNN	Deep Neural Network
ECT	Eddy Currents Test - Eddy Currents Testing\{}\{}
ECT	Eddy Currents Testing
FC-NN	Fully Connected Neural Networks
FEM	Finite Element Models
FFL	Focal Frequency Loss
FI	Feature Importance\{}\{}
FiLM	Feature-wise Linear Modulation
FMC	Full Matrix Capture
GAN	Generative Adversarial Network
GSA	Global Sensitivity Analysis
GT	Ground Truth
HF	High Fidelity
IA	Artificial Intelligence
ICA	Independent Component Analysis

IN	Instance Normalization
LF	Low Fidelity
LS	Latent Space (coordinates)
M-TFM	Multimode Total Focusing Method
MAE	Mean Absolute Error
MAPOD	Model-Assisted Probability of Detection
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
NDT&E	Non-Destructive Testing and Evaluation
NN	Neural Networks
PA	Phase Array
PCA	Principal Component Analysis
POD	Probability of Detection
ReLU	Rectified Linear Unit
ROI	Region of Interest
SA	Self-Attention
SAFE	Semi-analytical Finite Elements
SHM	Structural Health Monitoring
SL	Supervised Learning
SSL	Semi-Supervised Learning
ST	Spatial Transformer
SVM	Support Vector Machine
TFM	Total Focus Method
TL	Transfer learning
TTUR	Two Times-scale Update Rule

UL	Unsupervised Learning
UMAP	Uniform Manifold Approximation and Projection
UT	Ultrasound Test - Ultrasound Testing
VAE	Variational AutoEncoder
VIM	Volume Integral Method (VIM)

To my parents, my family and my friends.

They were a great pillar for me during this 3 years.

Acknowledgment

I extend my deepest gratitude to the members of the Thesis Committee for their unwavering support and invaluable feedback. Special thanks to Roberto Miorelli and Filippo Gatti, my dedicated supervisors, for their patience, enlightening feedback sessions, and the transfer of technical and scientific knowledge. The guidance and commitment from Didier Clouteau, our director, have been instrumental throughout this academic journey.

Heartfelt appreciation goes to my colleagues and friends from the laboratory for the interesting exchanges, occasional support, and the unforgettable afterwork and activities that have enriched this research experience.

A warm thank you to the CEA staff and CentraleSupélec lab staff for their welcoming and pleasant environment.

Jonathan and Germán, my closest friends and their lovely families, deserve a special mention. There are no words to express my gratitude for their constant presence, even across distances.

To Daniela, Laura, Kevin, Anita, Carla, Araceli, Macarena, Estefani, and Flor, my friends from the university and life. I am grateful for your unwavering support and always being there, no matter what.

Santi, Flor, Mica, Agus, Juli, Leti, my new friends, have been a source of joy with their company during mate sessions, asados, and parties that have made these three years more manageable.

A special note of appreciation to Tatiana, Francois, and Olek, my new friends, for their friendship and unique contributions, adding a special touch to my experience.

To Viviana and Fernando, my parents, your special support throughout my life, encouragement, and unwavering presence have brought me to this point. My love for you is unconditional.

Mery, Silvia, Domingo, and Lázaro, my grandparents, who were always proud of me. This achievement could also be a source of pride for them.

To Gabriel, Cande, and my cousins, my close family, who were always present when I was in my country.

Luis and Yani played a key role in facilitating my logistics to come to France during the challenging times of COVID-19, providing unconditional support.

Miriam and Pepe, my godmother and godfather, your love, support, and presence have been a source of strength.

A big shoutout to Marie, Nouhayla, André, Victor, Romain, Vivek, Clément, Rémi, Amine, Camille, Daniel, Alexandre, Jordan, Imanol, Lauren, Sabah, Vinduja, Amond, Valentin, Gottfried, Fanny and all CEA and CentraleSupélec colleagues. Your particular mentions in this list may not be exhaustive, but each one of you has played a significant role. From travels to bars, corridor discussions to shared meals in the canteen, the memories are cherished.

Luna, my loyal dog, deserves a special mention for showing unconditional love in both good and challenging moments.

To Angel, Virginia, Laura, Fernanda, Loïc, Kamal, some old colleagues from Argentina and France: your initial impulse in the research and continuous support have been invaluable.

I am grateful to Numeric program who funded this work and provided support during this process, and to my Numeric buddies for the engaging activities that kept us smiling throughout.

A sincere acknowledgment goes to Science Accueil and their dedicated staff for their role in facilitating the integration of a foreigner into a new academic and cultural setting. Your help has been invaluable.

To each and every person mentioned and those not explicitly named, your contributions, whether big or small, have played a crucial role in this academic endeavor. Thank you for being a part of this journey and for making it a truly enriching and memorable experience.

0 - Introduction

Context

In Non-Destructive Testing and Evaluation (NDT&E) and Structural Health Monitoring (SHM), simulation tools based on numerical models have been historically used to design and assess the performances of inspection methods and techniques. Some examples of simulation application are sensitivity studies, probe designing, acquisition interpretation, training, etc. More recently, simulations have also been exploited to produce the data needed to develop automatic diagnosis tools and algorithms. However, simulated data may be far from in situ data due to uncertainties linked to acquisition, post-processing, and operator factors. As a consequence, a systematic bias between simulation and experimental measurements leads to a poor generalization capability (i.e., the model performance once tested on experimental data) of the deployed models. On the other hand, using in situ data for several applications is often unpractical due to the lack of a sufficiently large number of annotated samples due to high costs associated to the annotation process. On top of that, the exploitation of NDT&E and SHM in situ data are often restricted to the companies that have taken the experimental measurements due to confidentiality issues.

In this context, the development of automatic diagnostic tools, e.g., Machine Learning (ML) diagnostic tools, Deep learning (DL) methods, or decision support systems, are developed either via simulated data or to very specific and case-dependent experimental data sets with restricted access. However, in both aforementioned cases, this introduces a bias that can lead to poor performance when these tools are finally deployed in the real world.

To tackle such an issue would consist in accounting the informative content embedded in both simulations and experimental data as much as possible. The informative content can be expressed in terms of *fidelity* levels. In this framework, fast and reliable simulation tools as applied to NDT&E and SHM can be considered low-fidelity data, while in situ collected data are considered as the highest fidelity data available. It is worth mentioning that other intermediate degrees of fidelity sources can be considered for NDT&E and SHM accordingly to the simulation tools employed or the inspection accounted. That is, output data from coarse and fine mesh Finite Element Models (FEM) models may be considered two different fidelity levels. Coarse mesh FEM models are faster but less faithful to reality, while fine mesh models are a more reliable approximation. Sources of fidelity levels are virtually infinite when considering different models, experimental data, data collection techniques (e.g., labeled or unlabeled data), and more. This work focuses on improving automatic diagnostics by considering and possibly blending different fidelity data available.

Deep learning (DL) methods have been successfully applied on NDT&E and SHM fields for simulation-driven inversion problems, particularly in ultrasound test imaging applications [1, 2, 3, 4, 5, 6, 7, 8]. The pursuit of efficient methods to enhance the automation of NDT&E and SHM inspections driven by artificial intelligence is a topic of increasing interest in the community [9, 10].

One of the challenges in DL is the need for a sufficiently large set of ‘reliable’ training data. On one hand, a large data set helps to avoid the ‘curse of dimensionality’ issue or ill-posed inverse problems. Conversely, exploiting training data close to the in situ data may lead to better convergence and generalization in the test phase on a given inversion task. Even if extensive simulated data set creation is nowadays possible and highly reliable simulations for NDT&E and SHM inspection are accessible, simulations barely reproduce the complexities of experimental or in situ data, which may embed spurious contributions due to the environmental and experimental conditions, human factors, etc.. To mitigate such a problem, a possible helpful scenario is to employ a multi-fidelity data set, when accessible, and combine simulations and experimental incertitude levels into an enhanced, and more extended in size, data set.

The natural approach when multiple fidelity levels are available is to try to use the more reliable data (highest fidelity level) as training data so the deployment of the automatic inspections reports a better performance. Unfortunately, in NDT&E and SHM problems, the highest fidelity data are not accessible, poorly labeled or very scarce (confidentiality, unknown data set production conditions, etc).

Recently, many methods have been developed to handle this problem in the field of NDT&E and SHM to take advantage of the full knowledge of this type of data set. Transfer learning (TL) is a common approach when a similar task (e.g., image feature extraction) has been performed previously on a different domain. These methods benefit from previously trained neural networks to facilitate training a new DL algorithm [11, 12, 13]. TL can be effective when some data from a fidelity level is present during the training to achieve better performance, even when the data set is far from the in situ data in terms of distribution and the data set is small compared to the complexity of the task. However, these approaches do not aim to exploit different fidelity levels when they are available.

A succeeding approach from TL is the Domain Adaptation (DA), suitable when more than one domain (or fidelity level) is available. However, when examining NDT&E as well as SHM data, a common observation is that the closer the data production is to in situ conditions, the less information is typically available regarding the data production process, as discussed in [14, 15]. This lack of information translates to a poorly labeled high-fidelity data set. DA strategies [16] have been tested in those scenarios to enhance DL algorithms for NDT&E and SHM inspections [17, 18, 19, 20]. To circumvent such an issue, DA proposed solutions can vary from extensive access of labeled data to just one fidelity level correctly labeled one. Many techniques applied in toy sets from the bibliography seem suitable for the massive realistic generation of NDT&E data. However, more DA approaches count only in enlarged data sets from the simulation without faithfully considering the realistic data.

Hence, this study involves developing a deep learning framework to exploit deep learning generative models as applied to NDT&E inspection problems targeting data generation and their exploitation for computationally intensive tasks (e.g., inversion, statistical studies, etc.). The final objective is a ML model (i.e., a surrogate model or metamodel) that learns to generate realistic data from simulation by considering non-simulated uncertainties or characteristics present on higher fidelity data (e.g., sources of noise). This framework is pretended to be used to generate new data from robust models. Later, the data are applied in automatic diagnostic tools on NDT&E development to improve its performance for a given task: material characterization, flaw detection, or characterization, among others.

This dissertation is structured as follow:

- In Chapter I, an overview of both NDT&E methods and techniques as well as an introduction to the machine learning methods and techniques is provided.
- Chapter II objective is to develop a *supervised* deep learning framework in a *simulated data set* to be used as a surrogate model for efficient statistical studies, knowing that the NDT&E simulation can be expensive in terms of time calculation and some diagnostic studies require extensive data to be assessed. This first approach is intended to asses whether DL methods are suitable for developing a surrogate model for NDT&E data targeting inspection problems that can be described with a high or very high number of parameters. The developed framework has permitted to be tailored for establishing a supervised learning multi-fidelity procedure.
- Chapter III objective is to develop a *supervised* generative approach based on *low-fidelity and high-fidelity data set* to generate new robust *multi-fidelity* data. The criteria respected during the data set production is to have a direct link between the two different fidelity levels. The first source is a simulation tool for an ultrasound testing application and the second is experimental data obtained from an equivalent mock-up. A tailored DL architecture is trained in a supervised way on the produced image data set. The labels are available in both simulation and experience, and the exact image mapping fidelity level into another level. A new multi-fidelity image data set is

generated as a result. The generated new data are evaluated and the limitations of the architecture are studied and discussed, particularly the need for a highly informed and labeled multi-fidelity data set.

- Chapter IV aims to develop a *partially non-supervised* DL approach for TFM *multi-fidelity data set* generation without prior information on the data coupling or labels. A new approach that does not rely on this prior information is preferred for the objectives of this thesis since many of the NDT&E data set do not count with this characteristic (e.g., poorly labeled data). The same data set in Chapter III is used to train a DL architecture, but this time by using a priori information on the domain (source or target) of the data employed. The architecture learns to generate different fidelity samples by a probabilistic approach. The generation capabilities of the architecture are evaluated. An inverse problem is assessed with the realistic data obtained from the DL surrogate model to prove its efficacy.

- Chapter V gives a summary of the presented work and some perspectives for the future.

Acknowledgments

The presented thesis was developed in the CEA LIST institute [1], an internationally recognized research institution in the field of nondestructive testing, together with the LMPS at CentraleSupélec[2].

CEA develops the CIVA [3] software, which offers multi-physics models and is among the leading products for simulation on NDT&E applications[21]. LMPS laboratory has developed multiple software in the domain of material study like OOFE[4] and develops artificial intelligence methods assisted by its computing center, Fusion/Ruche[5].

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 800945-NUMERICS-H2020-MSCA-COFUND-2017.

Publications

G.E. Granados, R. Miorelli, F. Gatti, S. Robert, D. Clouteau, **Towards a multi-fidelity deep learning framework for a fast and realistic generation of ultrasonic multi-modal Total Focusing Method images in complex geometries**, NDT & E International, Volume 139, 2023, 102906, ISSN 0963-8695, https://doi.org/10.1016/j.ndteint.2023.102906.

G.E. Granados, R. Miorelli, F. Gatti, D. Clouteau, **A deep learning framework for efficient global sensitivity analysis and SHAP values calculations applied to eddy current testing problems**, 50th Annual Review of Progress in Quantitative Nondestructive Evaluation 2023, Volume 87202, QNDE2023-118352, V001T10A001, 10 pages, https://doi.org/10.1115/QNDE2023-118352.

G.E. Granados, F. Gatti, R. Miorelli, S. Robert, D. Clouteau, **Generative domain-adapted adversarial auto-encoder model for enhanced ultrasonic imaging applications**, Engineering Applications of Artificial Intelligence, Submited.

[1]CEA LIST: LIST belongs to French Atomic Energy and Alternative Energies Commission (CEA) and is responsible for leading large-scale projects on advanced manufacturing, embedded systems, data intelligence and control of radiation for health.

[2]LMPS: Laboratory of Mechanics Paris-Saclay, CentraleSupélec, l'école normale supérieure and CNRS.

[3]CIVA is a platform developed for industrial applications and academic purposes based on semi-analytic numerical methods to simulate non destructive testing.

[4]OOFE, Object Oriented Finite Element: Object-oriented numerical platform written in C++ for finite element calculation of structures and materials, with more than 300 classes and 100,000 lines.

[5]https://mesocentre.universite-paris-saclay.fr/platforms/platforms.html

Synthèse en français

Dans le domaine du contrôle et de l'évaluation non destructifs (" Non-Destructive Testing and Evaluation ", NDT&E) et du contrôle de santé structurel (" Structural Health Monitoring ", SHM), les outils de simulation basés sur des modèles numériques ont été historiquement utilisés pour concevoir et évaluer les performances des méthodes et des techniques d'inspection. Les études de sensibilité, la conception des sondes, l'interprétation des acquisitions, la formation, etc. sont autant d'exemples d'applications de la simulation. Plus récemment, les simulations ont également été exploitées pour produire les données nécessaires au développement d'outils et d'algorithmes de diagnostic automatique. Cependant, les données simulées peuvent être très éloignées des données in situ en raison des incertitudes liées à l'acquisition, au post-traitement et aux facteurs liés à l'opérateur. Par conséquent, un biais systématique entre la simulation et les mesures expérimentales entraîne une faible capacité de généralisation (c'est-à-dire la performance du modèle une fois testé sur des données expérimentales) des modèles déployés. D'autre part, l'utilisation de données in situ pour plusieurs applications est souvent peu pratique en raison de l'absence d'un nombre suffisamment important d'échantillons annotés et des coûts élevés associés au processus d'annotation. En outre, l'exploitation des données NDT&E et SHM in situ est souvent limitée aux entreprises qui ont effectué les mesures expérimentales en raison de problèmes de confidentialité.

Dans ce contexte, le développement d'outils de diagnostic automatique, par exemple les outils de diagnostic par apprentissage automatique (" Machine Learning ", ML), les méthodes d'apprentissage profond (" Deep learning ", DL) ou les systèmes d'aide à la décision, se fait soit à l'aide de données simulées, soit à l'aide d'ensembles de données expérimentales très spécifiques et dépendantes du cas, dont l'accès est restreint. Cependant, dans les deux cas susmentionnés, cela introduit un biais qui peut conduire à de mauvaises performances lorsque ces outils sont finalement déployés dans le monde réel.

Pour résoudre ce problème, il faudrait tenir compte autant que possible du contenu informatif des simulations et des données expérimentales. Le contenu informatif peut être exprimé en termes de niveaux de fidélité. Dans ce cadre, les outils de simulation rapides et fiables appliqués au NDT&E et au SHM peuvent être considérés comme des données de faible fidélité, tandis que les données collectées in situ sont considérées comme les données de plus haute fidélité disponibles. Il convient de mentionner que d'autres degrés intermédiaires de sources de fidélité peuvent être envisagés pour la NDT&E et la SHM en fonction des outils de simulation utilisés ou de l'inspection comptabilisée. Ainsi, les données de sortie des modèles d'éléments finis ("Finite Element Models", FEM) à maillage grossier et fin peuvent être considérées comme deux niveaux de fidélité différents. Les modèles FEM à maillage grossier sont plus rapides mais moins fidèles à la réalité, tandis que les modèles à maillage fin constituent une approximation plus fiable. Les sources de niveaux de fidélité sont virtuellement infinies si l'on considère les différents modèles, les données expérimentales, les techniques de collecte de données (par exemple, les données étiquetées ou non étiquetées), etc. Ce travail se concentre sur l'amélioration des diagnostics automatiques en prenant en compte et éventuellement en mélangeant les différentes données de fidélité disponibles.

Les méthodes d'apprentissage profond ont été appliquées avec succès sur les champs NDT&E et SHM pour les problèmes d'inversion pilotés par simulation, en particulier dans les applications d'imagerie de test par ultrasons [1, 2, 3, 4, 5, 6, 7, 8]. La recherche de méthodes efficaces pour améliorer l'automatisation des inspections NDT&E et SHM pilotées par l'intelligence artificielle est un sujet qui intéresse de plus en plus la communauté [9, 10].

L'un des défis de l'intelligence artificielle est la nécessité de disposer d'un ensemble suffisamment important de données d'apprentissage "fiables". D'une part, un grand ensemble de données permet d'éviter la "malédiction de la dimensionnalité" ou les problèmes inverses mal posés. Inversement, l'exploitation de données d'entraînement proches des données in situ peut conduire à une meilleure convergence et à une meilleure généralisation dans la phase de test pour une tâche d'inversion donnée. Même s'il est aujourd'hui possible de créer de vastes ensembles de données simulées et que des simulations très fiables sont accessibles pour les inspections NDT&E et SHM, les simulations reproduisent

à peine les complexités des données expérimentales ou in situ, qui peuvent intégrer des contributions parasites dues aux conditions environnementales et expérimentales, aux facteurs humains, etc. Pour atténuer ce problème, un scénario utile possible consiste à utiliser un ensemble de données multi-fidélité, lorsqu'il est accessible, et à combiner les simulations et les niveaux d'incertitude expérimentaux dans un ensemble de données amélioré et de taille plus importante.

L'approche naturelle lorsque plusieurs niveaux de fidélité sont disponibles est d'essayer d'utiliser les données les plus fiables (niveau de fidélité le plus élevé) comme données d'entraînement afin que le déploiement des inspections automatiques donne de meilleurs résultats. Malheureusement, dans les problèmes de NDT&E et de SHM, les données les plus fidèles ne sont pas accessibles, sont mal étiquetées ou sont très rares (confidentialité, conditions de production des ensembles de données inconnues, etc.)

Récemment, de nombreuses méthodes ont été développées pour traiter ce problème dans le domaine du NDT&E et du SHM afin de tirer parti de l'ensemble des connaissances de ce type d'ensemble de données. L'apprentissage par transfert est une approche courante lorsqu'une tâche similaire (par exemple, l'extraction de caractéristiques d'une image) a été effectuée précédemment dans un domaine différent. Ces méthodes tirent parti des réseaux neuronaux précédemment formés pour faciliter la formation d'un nouvel algorithme d'apprentissage par transfert [11, 12, 13]. La TL peut être efficace lorsque certaines données d'un niveau de fidélité sont présentes pendant l'entraînement pour obtenir de meilleures performances, même si l'ensemble de données est éloigné des données in situ en termes de distribution et que l'ensemble de données est petit par rapport à la complexité de la tâche. Toutefois, ces approches ne visent pas à exploiter les différents niveaux de fidélité lorsqu'ils sont disponibles.

Une approche succédant à la TL est l'adaptation de domaine ("Domain Adaptation", DA), qui convient lorsque plusieurs domaines (ou niveaux de fidélité) sont disponibles. Cependant, lors de l'examen des données NDT&E et SHM, une observation commune est que plus la production de données est proche des conditions in situ, moins il y a d'informations disponibles concernant le processus de production de données, comme indiqué dans [14, 15]. Ce manque d'information se traduit par un jeu de données haute fidélité mal étiqueté. Les stratégies de DA [16] ont été testées dans ces scénarios pour améliorer les algorithmes de DL pour les inspections NDT&E et SHM [17, 18, 19, 20]. Pour contourner ce problème, les solutions proposées par la DA peuvent varier d'un accès étendu aux données étiquetées à un seul niveau de fidélité correctement étiqueté. De nombreuses techniques appliquées à des ensembles de jouets issus de la bibliographie semblent adaptées à la génération massive et réaliste de données NDT&E. Toutefois, un plus grand nombre d'approches d'évaluation quantitative ne prennent en compte que des ensembles de données élargis issus de la simulation sans tenir compte fidèlement des données réalistes.

Par conséquent, cette étude implique le développement d'un cadre d'apprentissage profond pour exploiter les modèles génératifs d'apprentissage profond appliqués aux problèmes d'inspection NDT&E ciblant la génération de données et leur exploitation pour des tâches à forte intensité de calcul (par exemple, inversion, études statistiques, etc.). L'objectif final est un modèle ML (c'est-à-dire un métamodèle) qui apprend à générer des données réalistes à partir de la simulation en tenant compte des incertitudes non simulées ou des caractéristiques présentes dans les données de plus haute fidélité (par exemple, les sources de bruit). Ce cadre est censé être utilisé pour générer de nouvelles données à partir de modèles robustes. Par la suite, les données sont appliquées dans des outils de diagnostic automatique sur le développement NDT&E afin d'améliorer ses performances pour une tâche donnée: caractérisation des matériaux, détection des défauts, ou caractérisation, entre autres.

Cette thèse est structurée comme suit :

- Dans le chapitre I, une vue d'ensemble des deux méthodes et techniques de NDT&E ainsi qu'une introduction aux méthodes et techniques d'apprentissage automatique sont fournies.

- L'objectif du chapitre II est de développer un cadre d'apprentissage profond supervisé dans un

ensemble de données simulées à utiliser comme modèle de substitution pour des études statistiques efficaces, sachant que la simulation NDT&E peut être coûteuse en termes de calcul de temps et que certaines études de diagnostic nécessitent des données étendues pour être évaluées. Cette première approche vise à évaluer si les méthodes DL sont adaptées au développement d'un modèle de substitution pour les données NDT&E ciblant les problèmes d'inspection qui peuvent être décrits avec un nombre élevé ou très élevé de paramètres. Le cadre développé a permis d'être adapté à l'établissement d'une procédure d'apprentissage supervisé multi-fidélité.

- L'objectif du chapitre III est de développer une approche générative supervisée basée sur des ensembles de données de basse fidélité et de haute fidélité afin de générer de nouvelles données multi-fidélité robustes. Le critère respecté lors de la production de l'ensemble des données est d'avoir un lien direct entre les deux différents niveaux de fidélité. La première source est un outil de simulation pour une application de test par ultrasons et la seconde est constituée de données expérimentales obtenues à partir d'une maquette équivalente. Une architecture DL personnalisée est entraînée de manière supervisée sur l'ensemble des données d'images produites. Les étiquettes sont disponibles à la fois dans la simulation et dans l'expérience, et le niveau de fidélité exact de la cartographie d'image dans un autre niveau. Un nouvel ensemble de données d'images multi-fidélité est ainsi généré. Les nouvelles données générées sont évaluées et les limites de l'architecture sont étudiées et discutées, en particulier la nécessité d'un ensemble de données multi-fidélité hautement informées et étiquetées.

- Le chapitre IV vise à développer une approche DL partiellement non-supervisée pour la génération d'ensembles de données multifidélité TFM sans information préalable sur le couplage des données ou les étiquettes. Une nouvelle approche qui ne repose pas sur ces informations préalables est préférable pour les objectifs de cette thèse car de nombreux ensembles de données NDT&E ne présentent pas cette caractéristique (par exemple, des données mal étiquetées). Le même ensemble de données du chapitre III est utilisé pour former une architecture DL, mais cette fois en utilisant des informations a priori sur le domaine (source ou cible) des données employées. L'architecture apprend à générer des échantillons de fidélité différente par une approche probabiliste. Les capacités de génération de l'architecture sont évaluées. Un problème inverse est évalué avec les données réalistes obtenues à partir du modèle de substitution DL pour prouver son efficacité.

- Le chapitre V présente un résumé des travaux présentés et quelques perspectives pour l'avenir.

La thèse présentée a été développée au sein de l'institut CEA LIST, qui appartient au Commissariat à l'énergie atomique et aux énergies alternatives (CEA) et est responsable de la conduite de grands projets sur la fabrication avancée, les systèmes embarqués, l'intelligence des données et le contrôle des rayonnements pour la santé, une institution de recherche internationalement reconnue dans le domaine des essais non destructifs, avec le Laboratoire de mécanique Paris-Saclay (LMPS) de Centrale-Supélec, l'École normale supérieure (ENS) et le Centre national de la recherche scientifique (CNRS).

Le CEA développe le logiciel CIVA qui est une plateforme développée pour des applications industrielles et à des fins académiques basée sur des méthodes numériques semi-analytiques pour simuler des essais non destructifs. Le logiciel CIVA offre des modèles multi-physiques et est parmi les produits leaders pour la simulation sur les applications NDT&E. Le laboratoire LMPS a développé de nombreux logiciels dans le domaine de l'étude des matériaux, comme OOFE (Object Oriented Finite Element) : Plate-forme numérique orientée objet écrite en C++ pour le calcul par éléments finis des structures et des matériaux, avec plus de 300 classes et 100 000 lignes. et développe des méthodes d'intelligence artificielle avec l'aide de son centre de calcul, Fusion/Ruche.

I - An overview of machine learning frameworks for blending numerical simulations and experimental data as applied to NDT&E and SHM problems

About this section

This chapter introduces the theoretical frame related to the aimed non-destructive test and evaluation methods. Simulation approaches are discussed toward the strategy and methods definition. Some concepts and techniques from the bibliography are presented as an introduction to machine learning. They are the candidates to be applied in realistic simulations in non-destructive testing. The state of art on forward models on numerical simulation and inversion techniques assisted by machine learning and some first approaches are introduced.

I.1 . Assessing material integrity and properties via non-destructive test and evaluation and structural health monitoring

Non-Destructive Testing and Evaluation (NDT&E) and Structural Health Monitoring (SHM) methods are widely used in industry to assess the state of a material without damaging it. These methods aim to study a critical component of a system by keeping its shape and internal structure as it is. This is possible by observing the interaction of a specimen with a probing source such as sound, ultrasound and electromagnetic wave propagation, radiation and absorption, responses. NDT&E and SHM intend to infer the state of its structure on the studied specimen by interpreting the interaction results. Some application areas are the nuclear, metallurgic, aeronautic, and aerospace industries. Furthermore, mores specifically in the case of SHM-based methods, remaining useful life and properties degradation estimation allows users to make decisions before a component fails. In contrast, the lifetime is increased thanks to preventive or predictive maintenance.
Provided a given inspection method, different techniques can be specifically employed to infer properties of the material (material characterization), such as the microstructure achieved in the additive fabrication process, detection and sizing of cracks, corrosion, or material proprieties degradation.

To categorize the different NDT&E methods, one can classify by the energy source to probe the object, the nature of the signals resulting from interaction, and means of detecting the resulting signal, among others. Toward this end, among the most relevant NDT&E methods, one can cite:

- Visual testing, liquid penetrant testing, and magnetic particle testing methods are local inspections that require qualified operators and are not often automated.
- Ultrasonic Testing (UT) uses high-frequency sound waves to detect defects in materials. UT works by transmitting a pulse of sound waves into the material and then measuring the reflected echoes. The time it takes for the echoes to return to the transducer can be used to determine the distance to the defect, and the amplitude of the echoes can be used to determine the size of the defect. UT is a versatile method that can be used to inspect a wide range of materials, including metals, plastics, and composites. UT is commonly used in the aerospace, automotive, manufacturing, oil and gas, power generation, and welding industries.
- Eddy Current Testing (ECT) uses an alternating magnetic field to detect defects in conductive materials. ECT works by inducing eddy currents in the material and then measuring the changes in the eddy currents caused by defects. ECT is a sensitive method that can be used to detect a variety of defects, including cracks, corrosion, and heat damage. ECT is commonly used in the aerospace, automotive, electronics, manufacturing, and power generation industries.
- Radiographic Testing (RT) uses X-rays or gamma rays to produce images of internal defects in

materials. RT works by passing a beam of X-rays or gamma rays through the material and then detecting the X-rays or gamma rays that pass through the material. A possible interpretation of the resulting interaction of the matter with the X-rays, the areas of the material that are less dense will allow more X-rays or gamma rays to pass through, resulting in a darker image on the film or digital detector. RT is a versatile method that can be used to inspect a wide range of materials, including metals, plastics, and composites. RT is commonly used in the aerospace, automotive, manufacturing, oil and gas, power generation, and welding industries.

- Infrared Thermography (IT) uses infrared cameras to detect differences in temperature on the surface of a material. IT works by measuring the amount of infrared radiation emitted by the material. The areas of the material that are hotter will emit more infrared radiation, resulting in a brighter image on the infrared camera. IT is a sensitive method that can be used to detect a variety of defects, including hot spots, corrosion, and delamination. IT is commonly used in the aerospace, automotive, construction, electrical, and manufacturing industries.

- Acoustic Emission Testing (EAT) uses transducers to detect elastic stress waves generated by crack propagation. EAT works by attaching transducers to the surface of the material and then registering the elastic stress waves. EAT is a sensitive method that can be used to detect cracks and other defects in a variety of materials, including metals, plastics, and composites. EAT is commonly used in the aerospace, automotive, manufacturing, oil and gas, and power generation industries.

- Structural Health Monitoring (SHM) uses a variety of sensors to monitor the health of a structure. SHM works by attaching sensors to the structure and then monitoring the data from the sensors for changes. SHM often uses guided waves (GW) across a plate or pipe. Guided waves can be used to inspect structures for defects by generating guided waves at one location and then measuring the guided waves at another location. SHM can be used to detect a variety of defects, including cracks, corrosion, and fatigue. SHM is commonly used in the aerospace, bridges, buildings, and dams industries.

The following sections provide a gentle introduction to UT and ECT since these inspection data have been employed for assessing the performance of the ML methodology developed in this thesis. It is worth mentioning that the current accessibility to data justifies the choice of these techniques, but the methodologies deployed on them are not restricted only to UT or ECT data and they may applied, up to some extent, straightforwardly to other NDT&E and SHM methods and techniques (e.g., SHM guided wave imaging, thermographic testing).

I.2 . Advanced simulation tools in NDT&E and SHM

The increasing computational power has led to a big leap forward in the simulations capabilities on both desktop computers and grid/cloud computing. Therefore, the realism and the complexity of NDT&E and SHM have sensibly increased in the last decade. That is, forward simulation on NDT&E and SHM (Fig. I.1) proved to be an extensive tool to study different failure cases justified by the possibility of varying parameters set-up like crack position, material properties, or sensors path on demand. This capability is habitually exploited for:

1. Interpretation of acquisition: recorded data needs to be interpreted. Since it is an indirect measurement of, e.g., a defect or material property, the time series and images obtained in the experience are not directly informative about the final objective of the inspection. Forward simulations help to understand how to post-process the raw data.

2. Design probes and acquisition setup: the more suitable setup and probe are searched. Once the physics of the interaction between the transducer and the specimen is known, finding the best

configuration to find flaws or defaults or even to characterize the material is possible by simulating the experience in different configurations and evaluating its performance.

3. Evaluate the performance of an inspection setup: the capability of detection, sensibility to uncertainties, and measurement errors can be evaluated by varying the parameters of a configuration on the forward simulation. Different sources of error can be used to understand how a variation in an inspection parameter can affect its performance.

4. Generate databases to perform statistical studies: The possibility of creating significant amounts of data and examples by forward modeling proved useful for performing statistics studies on the implied parameters of an inspection. Some examples are quantifying reliability, probability of detection, and comparison between techniques.

5. Diagnostic tools and decision support systems tuning before its deployment on real-world data: forward simulation can provide a means to produce the number of examples that are needed for inverse problems (Fig. I.1). Since an inverse problem uses a large amount of data, the cost of production of a numerical simulation presents a solution against the costly experimental acquisition. New approaches like machine learning models solutions to inverse problems exploit this.

The numerical simulation approach requires a thorough physical understanding of the process to derive analytical models that closely describe the interaction with the material. Modeling often relies on Finite Element Methods (FEM), which describe the material in small discrete regions (mesh) by computing the partial differential equations in space (2D or 3D). This method makes it possible to describe complex geometries and material micro-structures considering local effects. Complex numerical models can add reliability and improved simulation by adding, e.g., structural noise with some constraints [22][23], but at the same time, they increment the calculation cost.

Conversely, semi-analytical modeling relies on an approximated microstructure representation, giving a faster solution. This approach approximates analytical theories using assumptions that work well in, e.g., isotropic materials. However, these approximations may fail at predicting results in some complex configurations, e.g., in a flawed region.

Therefore, hybrid models, Semi-analytical Finite Elements (SAFE), combine semi-analytical and FEM methods. SAFE allows quickly describing most regions in a material with analytical approximation and complex interaction in a particular region of interest. Together, they combine the advantages of both methods while minimizing their inconveniences [24].

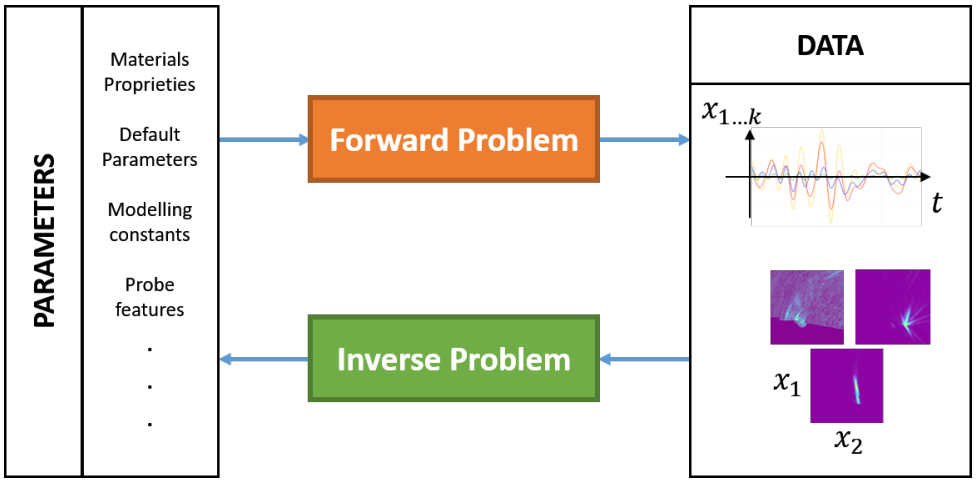


Figure I.1. Forward and inverse problem schema on NDT&E. Signals represent a generalized case of a multiple signal $[x_1, ..., x_k]$ on time or a 2D image $[x_1, x_2]$ acquisition on UT.

In addition, inverse problems on NDT&E solved by closed-form methods also demand a wide knowledge of physics and model parameters. Diagnostic tools conceived for NDT&E techniques and calibrated

on simulation data present a lower performance on in situ data once they are deployed in situ. This performance gap is mainly due to the difference (e.g., uncertainties) between the simulated and real-world data.

Meanwhile, data-driven approaches to diagnostic tools based on heuristics compete for on-time execution and accuracy with closed-form methods approaches on NDT&E applications [25][26][27]. The main challenge of these approaches is accessing a large amount of data from where the algorithm can learn.

Additional assumptions for the optimization problem are needed to avoid over-fitting or instability issues. In the frame of this thesis, costly data like experimental records are poorly available, and labeled data of this nature is even rarer. The inverse problem approach here deals with both realities: synthetic data are not good enough for regression or classification problems, and realistic data are not enough.
Fast forward solvers available are a fair solution to produce many examples, but they are still inaccurate compared to the actual inspection case. CIVA simulations, as an example, can provide data production to evaluate diagnostic tools on non-destructive fields. Nevertheless, simulated signals inherently look 'perfect' and are, for instance, easily distinguishable from experimental data by any expert in the field and classified as 'synthetic' based on their appearance.

Consequently, diagnostic tools based on ML have the problem of being trained with simulated data that do not represent the experimental cases. A new simulation approach can be foreseen to produce simulated data that can be considered close to reality to enhance the performance of the mentioned tools.

The next Sections describe simulation principles of some NDT&E methods to provide the basis for experimental and synthetic (simulated) data set production.

I.3 . An overview of eddy current testing inspection techniques and simulations

ECT is a NDT&E method to explore a material property or internal structure using the electromagnetic interaction between probing source and material properties changes. In detail, electromagnetic induction is used to generate eddy currents in a conductive specimen (an example of an inspection of a plate is shown in Fig. I.2). The resulting induced magnetic field is measured and interpreted to detect and localize defects or thickness changes in the material or even to characterize it, among others.

ECT is a non-contact and non-ionizing method suitable for detecting discontinuities at the surface or below the surface of the specimen. For this, a probe commonly constituted by one or many copper wire coils generates an oscillating magnetic field when alternating current flows. The standard frequency range goes from about a few hundred hertz to a maximum of 10 MHz.

There exist almost as many coil configurations as existing ECT applications (e.g., green ring on Fig I.2). Some configurations are manual, while some are adapted to automatic inspection procedures. The relatively simple manufacturing of ECT probes allows the design of case-adapted shapes and configurations. For instance, surface and ring probes are suitable for crack detection in surfaces, while rotating scanner probes are adapted to large-diameter hole inspection. Axial probes, for instance, may be used for tube inspections. The aspects to consider in the design may encompass inductance and resistance of the coil; distribution of the magnetic field; coil sensibility to changes in material properties; characteristics related to the distance from the target (lift-off); response to notches, drilled holes, or other irregularities. Furthermore, the design might be influenced by various constraints intrinsic to the testing environment, such as weather conditions or access requirements for specific shapes or sizes. The design process typically involves iterations and progresses through trial and error.

ECT techniques can also vary regarding how the probe measures the observed magnitude changes. Probes typically operate in one of four fundamental modes: absolute, differential, reflection, or hybrid. The absolute mode uses one coil to measure the magnetic field changes. The differential mode uses two coils wound in opposition, where any difference in the magnetic field may indicate that one of the coils is in the presence of a flaw. The differential mode uses driver/pickup probes where one coil excites the eddy currents, and the other is used to sense changes in the test material. Hybrid coils implement a combination of the other modes.

The benefit of the ECT method is the inspection speed compared to other NDT&E methods, being one of the quicker inspection procedures. The cost is also competitive with the rest of NDT&E methods, together with the portability and versatility of the probes. Some known limitation of ECT is its susceptibility to permeability changes on the material, the only application to conductive materials, the low detention of parallel flaws, and its non-suitability for large inspection areas. A central issue is the limited penetration depth, where UT techniques compete with ECT techniques.

ECT operation principles In ECT inspections, every coil is characterized by the impedance parameter Z_0, which is a complex number defined as in Eq. I.1, and it represents the voltage-current ratio (V_0/I_0) for a single frequency sinusoidal excitation f. Impedance Z_0 has a magnitude $|Z|$ and a phase φ [28]:

$$Z_0 = \frac{V_0}{I_0} = R_0 + jX_0 = \sqrt{R_0^2 + X_0^2}_{\varphi=\arctan 2(X_0/R_0)} = |Z|_\varphi \tag{I.1}$$

When an alternating current energizes a coil, it creates a time-varying magnetic field. The magnetic flux lines tend to be concentrated at the center of the coil. Eddy current inspection is based on Faraday's electromagnetic induction law as demonstrated in Eq. I.2. The electromotive force ε is proportional to the time-rate change of the magnetic induction flux density Φ_B

$$\varepsilon = -\frac{d\Phi_B}{dt} \tag{I.2}$$

When an alternating energized coil of impedance Z_0 approaches an electrically conductive non-ferromagnetic material in a pristine specimen, a primary alternating magnetic field generated by the coil penetrates the material and it generates continuous and circular eddy currents on the specimen. The induced currents flowing within the test specimen generate a secondary magnetic field that tends to oppose the primary magnetic field. This opposing magnetic field, coming from the conductive material, weakens the primary magnetic field. In effect, the new imaginary part X_c of the coil impedance decreases proportionally when the eddy current intensity in the test piece increases. Eddy currents also increase the coil energy power dissipation that changes the real part R_c of coil impedance. Measuring this coil impedance variation can be infered by

$$\Delta Z = Z_c - Z_0 = (R_c - R_0) + j(X_c - X_0) \tag{I.3}$$

by monitoring either the voltage or the current signal can reveal specific information such as conductivity and chemical composition of the test piece.

In application where the material may present differences in thickness or a flaw in its surface, a $2D$ mapping image can be generate by monitoring, as instance, $|\Delta V|$ at different positions coil at the specimen surface. The resulting image can be used for discontinuities localization. Impedance amplitude and phase angle changes can be used to identify discontinuities, as shown in Fig. I.2.

I.3.1 . ECT parametric simulation on CIVA for data production

ECT inspection simulation typically focuses on the probe signal associated with a specific specimen discontinuity. When different conditions are considered for the inspection, parametric studies can be helpful in coil designing and setup optimization. Identifying discontinuities signatures may also be an objective when simulating ECT. Differential equations that govern the flaw interaction phenomena are derived from Maxwell's equation. For instance, the CIVA semi-analytical model with the Volume Integral Method (VIM) is based on Green's formalism [29, 30, 31] for canonical geometries. Other solvers based on finite integration techniques or surface integration equations are also deployed for more complex geometries.

The VIM formulation proposes the solution of an integral equation in different stages. The computation of the pristine material field $\mathbf{E}_\mathrm{p}$ is the first stage, known as the primary field and computed by the Method of Moments, a full-wave numerical method that discretizes the integral equations of electromagnetic fields [32]. Then, the flaws are modeled as a set of fictitious sources. So, the interaction between the flaws and the primary electric field (Eq. I.4) can be expressed as,

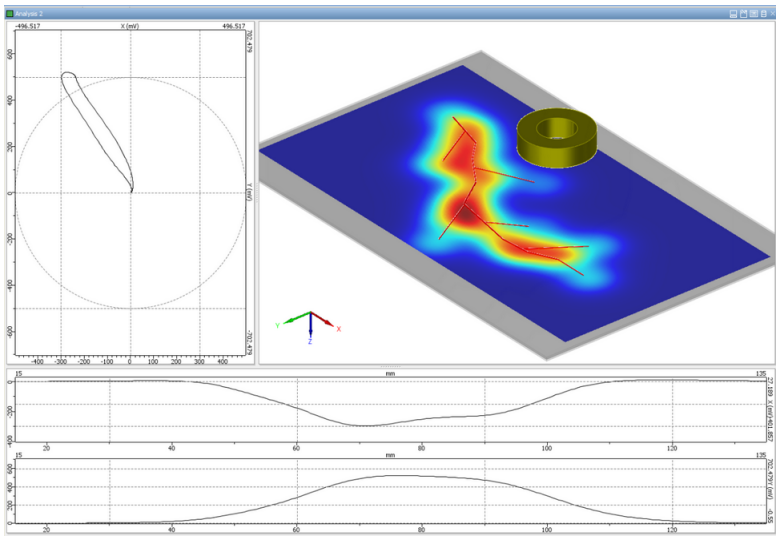


Figure I.2. Simplified 3D schema of ECT inspection configuration for defect characterization on metallic specimen. Impedance changes for a coil displacement at the material surface (x, y) are shown in the three plots. Changes in phase and amplitude are observed. $|\Delta V|$ versus coil position can produce a 2D surface mapping to observe the flaw regions, as shown in the 3D schema. Source: http://www.extende.com

$$\mathbf{E}_{\mathrm{p}}(\mathbf{r}) = \mathbf{E}_{\mathrm{t}}(\mathbf{r}) + i\omega\mu_0 \int_{\Omega} \mathbf{G}(\mathbf{r}, \mathbf{r}')[\sigma_0 - \sigma(\mathbf{r}')]\mathbf{E}_{\mathrm{t}}(\mathbf{r}')\, \mathrm{d}\mathbf{r}', \tag{I.4}$$

where Ω is the volume of the flaw, ω is the angular frequency, $\mu 0 = 4\pi.10 - 7$, G is a dyadic Green's function, σ_0 is the material conductivity, $\mathbf{E}_{\mathrm{t}}(\mathbf{r})$ is an exciting term due to the probe. $\sigma(\mathbf{r}')$ is the flawed region of conductivity.

The computation of the total field $\mathbf{E}_{\mathrm{t}}$ is then used to find the ECT signal (coil response) with the reciprocity theorem,

$$\Delta V = \frac{1}{I} \int_{\Omega} [\sigma_0 - \sigma(\mathbf{r})]\mathbf{E}_{\mathrm{t}}(\mathbf{r}) \cdot \mathbf{E}_{\mathrm{p}}(\mathbf{r})\mathrm{d}\mathbf{r}, \tag{I.5}$$

where ΔV is the voltage change in the coil by the presence of a flaw, and I is the coil current. The formulation in Eq. I.5 shows how to model the interaction of a flaw with the coil so the quantities in Fig. II.3 can be computed. ΔV is affected by some parameters besides the flaw geometry as, for instance, relative coil position, or specimen conductivity and thickness.

Given this general formulation, it is worth noting that a more suitable formulation exists for specific inspection cases. For example, [33], [34] and [35] present a boundary element method formulation for a planar-stratified conductive medium damaged by one or several narrow cracks.

I.4 . An overview of ultrasound testing inspection techniques and simulations

Ultrasound testing is a NDT&E method to explore a material property or internal structure by probing the medium under testing via an ultrasonic wave source. The frequency of mechanical wave propagation through solid, liquid, or gas, or a mix of them in some applications, is in the range of 20 kHz to 1 GHz. Similarly to other methods, sound waves interact with the medium by reflection, refraction, attenuation, and diffusion.

Two types of body waves are commonly considered on emission and recording during an UT inspection. Longitudinal waves (L) produce a matter displacement through compression through the propagation axis. On the other hand, traversal waves (T), also known as shear waves (denoted by S too), displace

the particles of the medium in the transverse direction of propagation (Fig. I.3). During interactions in a material, multiple conversions from L-waves to T-waves or vice-versa happen. Surface waves (also known as Rayleigh waves) affect UT inspections but are not described here. These waves mix L and T particle motions and are recorded on guided wave applications for long-distance inspection, together with the body waves.

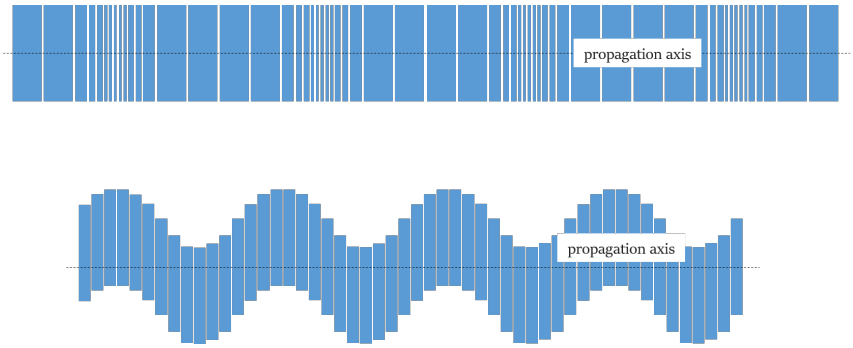


Figure I.3. 2D view of Longitudinal (up) and Transverse (down) wave propagation and particle displacement.

The sound waves are produced near the specimen (in contact with the surface or distant source) with piezoelectric crystals on active techniques. In another sense, a material on service may emit ultrasonic waves on its own (e.g., pressure vessels, pipelines, storage tanks, aircraft structures, or steel cables). The techniques that used this acoustics emission are referred to as passive. The current section will describe UT only on metallic materials applications.

Ultrasound testing techniques typically record the wave echos propagated on the material with the piezoelectric transducer in contact with the surface. Different configurations for varied applications exist: mono emitter/receptor, dual transducer (emitter and receptor in one probe), one or two Phase Array (PA) with wedges (emitter/receiver together or separated), and other possible contact configurations. A simplified UT technique configuration for defect characterization is shown in Fig.I.4 for complex geometry in a welded joint.

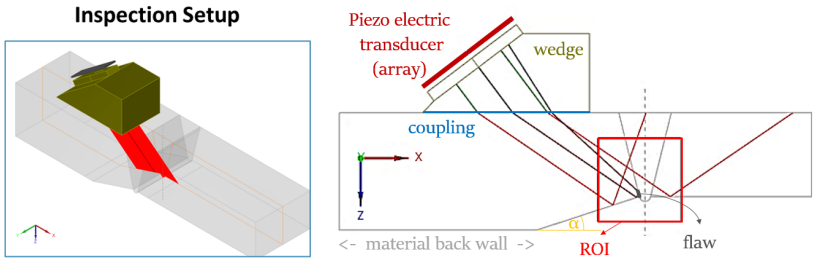


Figure I.4. Simplified schema of UT inspection configuration for defect characterization on a metallic specimen. The wavefront path emitted by one of the PA elements (red square) and echoes from flaw interaction captured by other elements is represented by red lines. A region of interest is define, where the ultrasound wave interacts with a possible flaw. A $2D$ Region of Interest (ROI) is selected around the area where a flaw may be present (red square).

Phase array transducers are particularly useful for imaging techniques on UT. Since those techniques allow the recognition of patterns linked to geometry and default localization, these probes are widely applied to material inspection. The device has many piezoelectric transducers that act as emitters and receptors alternately. Different arranges of the piezoelectric disposition over the wedge give advantages to specific material geometries: linear, annular, matrix, or circular. Post-processing of these data are

usually required to interpret the acquisition. A particular mention has to be made for the most recent UT inspection techniques like Laser Ultrasound (LUS) where the interaction with an impinging laser beam and the medium are probed via laser excitation and interferometry, respectively.

Particular applications of UT can be found in internal discontinuity detection, cross-section evaluation, bond characterization, or internal physical properties such as grain size, elastic constant, or structure. Compared to some hand-made NDT&E methods, UT offers an economical means for high-speed, large, or small-scale testing of materials. This method provides highly sensitive and accurate estimation of discontinuities by manual or automated experiences. Some techniques are limited by component geometry, internal structure characteristics, and coupling between the source and the material surface, and their compatibility. Surface rugosity, specimen temperature and quality of the coupling between probe and inspected medium can also be a limitation or source of uncertainty while testing a material.

UT operation principles In UT inspections, an ultrasonic transducer emits an ultrasonic wave that propagates in the component being inspected and is scattered by heterogeneity such as defects and eventually gives rise to echoes detected by a transducer acting as a receiver (often the same transducer acts as an emitter and receptor). Different approaches can be found to simulate UT inspections. Regarding CIVA simulation, the method used is the computation of retro-propagated energy in each point of the volume in the study, together with a high-order spectral finite elements approach in the defect area. This semi-parametric approach is called a coupled model.

For the computation of the retro-propagated energy in each point of the volume, waves are described by a full incident beam approach. Beams are considered as a set of rays with various times of flight and directions. For each location considered at the volume, the beams of both the emitting and receiving probes are considered to be sets of rays. The model is based on transposing the electromagnetic wave theory to electrodynamic waves to predict how a narrow beam or 'pencil' (Fig.I.5) of rays propagates from a source point to a computation point, this is called beam computation. A pencil is a mathematical object associated with an axial ray that follows the geometric energy path for a given wave propagation mode. A set of rays deviates infinitesimally from the axial ray. The term 'paraxial ray' is used for a ray belonging to the pencil envelope.

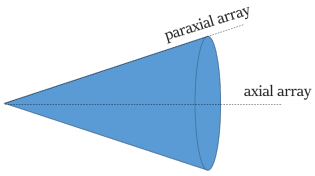


Figure I.5. Schematic description of a pencil in terms of axial and paraxial rays

Beam computation usually takes place in two stages. In stage one, the different energy paths are determined according to Snell-Descartes laws, generalized to anisotropic media. In stage two, the energy associated with each path is quantified. The conservation of energy principle is applied, assuming that energy flux (which follows the axial ray) is contained inside the pencil envelope. It can thus be deduced that acoustic intensity decreases (or increases) with pencil cross-section.

Modeling involves taking into account the following points [21]:

1. Emission of the wave and its propagation from the transmitter to the area being inspected,
2. Interaction (scattering) with acoustic discontinuities within this area (defects or boundaries),
3. Propagation to the receiver and reception.
4. Attenuation and structural noise (simplified approach only for some materials)

The calculation of the field radiated by an arbitrary transducer into a specimen is based on an extended form of Rayleigh integral, taking into account refractions and reflections at interfaces constituting the component to be tested. The particle displacement at point P, $\mathbf{u}_P(t)$ is written as [36]:

$$\mathbf{u}_P(t) = \mathbf{V}(t) \otimes \int\int \mathbf{G}_{SP}(t)\, dS \tag{I.6}$$

where $\mathbf{G}_{SP}(t)$ is the transient Green function for a point source S and an observation point P, and $\mathbf{V}(t)$ is the particle velocity perpendicular to the emitting surface. The $\otimes$ is defined as the tensor product. DyeThe elementary field $\mathbf{G}_{SP}(t)$ for each couple of points of the volume is evaluated analytically by means of the pencil method approximation. Each elementary field $\mathbf{G}_{SP}(t)$ can be expressed by means of a time and an amplitude, which are evaluated along the geometrical ray path between S and P.

The amplitude is evaluated using the divergence of a cone of rays (i.e. a pencil) surrounding the geometrical path. Its evolution depends on the media and interfaces crossed by the pencil, and is mathematically described by a product of matrices. This method allows to take into account anisotropic and heterogeneous components. Since the integration over the emitting surface is performed numerically (the surface being discretized into point sources), the transducers considered can also be arbitrarily shaped, including phased array transducers. The resulting field can be expressed in terms of impulse responses, when $\mathbf{V}(t)$ is a Dirac δ-function.

I.4.1 . UT parametric simulation on CIVA for data production

In order to consider flaw in the specimen, the $3D$ UT CIVA simulation applies a variant of the SAFE coupling models [37, 38] to approximate the response of a default in a studied $3D$ specimen. In this case, rather than constituting a generic FEM solver based upon a geometrical mesh of the defect (or of its surrounding), a lexicon of parametric defects is built. Each parametric defect is associated with a composition of geometrical structures referred to as macro-elements. In essence, a macro-element is defined as a potentially nonlinear deformation of a reference cube, and the complete set of macro-elements are arranged altogether in order to fit the defect geometrical description (Fig. I.6). The reference cube also bears a predefined hexahedral mesh, so that a macro-element inherits from this mesh through its associated deformation.

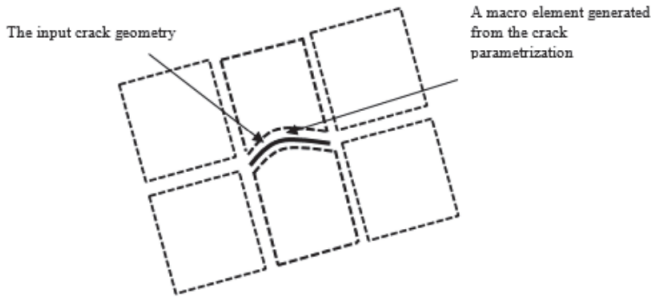


Figure I.6. Illustration of the macro elements arranged around an embedded crack for 3D UT simulation from [38].

The input data of the high-order spectral finite elements numerical model is obtained in the surroundings of the defect using the ray based beam computation model in the healthy part (Eq. I.6). The elastodynamic response in the presence of a flaw is then obtained by applying the Auld's principle of reciprocity. In harmonic regime, for a spectrum of the input signal S_i, the elastodynamic response spectrum S_e of a defect with a surface Σ is expressed as

$$S_e = S_i \int_{\Sigma} (\mathbf{u}_E \cdot \mathbf{T}_R - \mathbf{u}_R \cdot \mathbf{T}_E)\, d\Sigma, \tag{I.7}$$

where $\mathbf{u}$ and $\mathbf{T}$ denote the displacement vector and the normal stress tensor of the harmonic fields. E index denotes the ultrasonic field radiated by the emitting probe in the healthy component, and R index

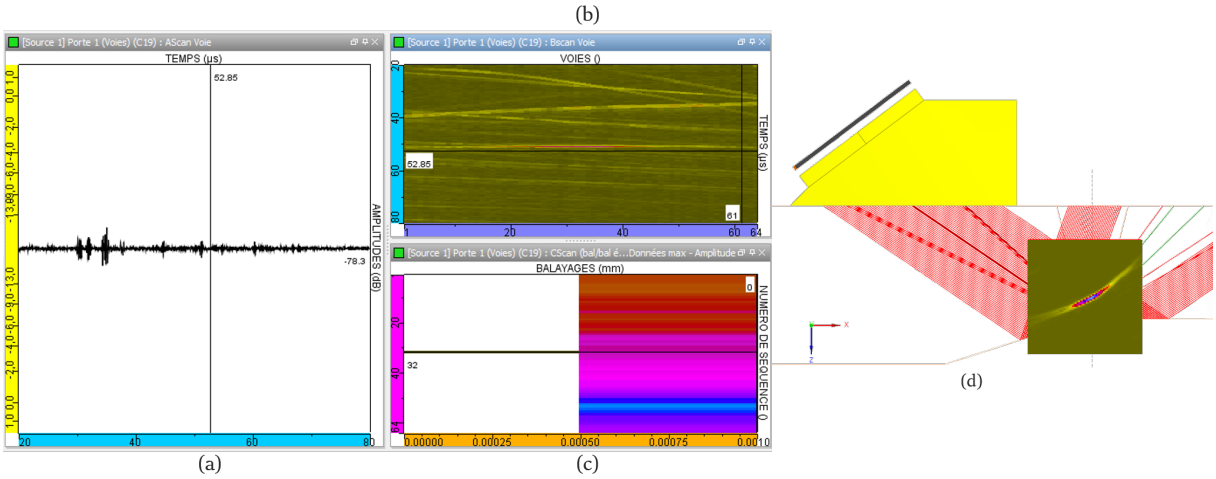


Figure I.7. Different data acquisition techniques from the UT method on the same specimen and post-processed example (TFM). Examples of (a) A-Scan of wave emission and noise recording, (b) B-Scan image phase array, and (c) C-Scan image (d) TFM of butt weld with a flaw in the material interface, the image reconstruction is shown in the region of interest. Source: http://www.extende.com

denotes the ultrasonic field radiated by the reception sensor (used as a fictitious emitter) in the presence of the defect.

If S_e is time-dependent, the Eq. I.7 can be rewritten as

$$S_e(t) = S_i(t) \int \int_{\Sigma} (\mathbf{u}_E(\tau) \cdot \mathbf{T}_R(t-\tau) - \mathbf{u}_R(t-\tau) \cdot \mathbf{T}_E(\tau))\, d\Sigma\, d\tau, \tag{I.8}$$

$S_e(t)$ can be bent into a time-depend quantity to be measured by the receiver. As an example, displacement or pressure amplitude $A(t)$ in the specimen surface can be recorded by a single emitter-receiver probe to generate a signal, as shown in Fig. I.7(a), called A-Scan.

UT techniques can be separated by acquired data strategy (Fig. I.7). Acquisition data strategy depends too in the type of probe (single probe or phase array (PA). Some of them are:

1. A-scan technique registers the amplitude of the wave versus time (Fig. I.7(a)). When the probe is static with one emitter and one receiver, the A-scan shows the emission moment and the echo delay received. This information helps the operator infer the thickness in the position and find defects echoes in a damaged section. This technique shows the function of amplitude A versus time t, $A(t)$. For PA probes, an A-scan is one couple emission-reception of each couple of the probe. An acquisition for a PA produces multiple A-scan signals.

2. B-scan is more informative than A-scan. It uses the A-Scan obtained in different probe positions or a PA emission-reception couples (e_{ij}) to produce images corresponding to the inspected area (cross-section of the material), named by Region of Interest (ROI). The high amplitudes in the graph are related to a defect in the area or echoes from the back wall. This technique shows the 2D function of the amplitude and time $A(t)$ at each position x or emission-reception couples e_{ij}, $A(x,t)$ or $A(e_{ij},t)$ in Fig. I.7(b).

3. The C-scan output is an image when A-scans are recorded in different emission-reception combinations using a fixed PA probe. The emission and reception sequences can varied by choosing different combinations of transducers used for A-Scan. In case of one element emit, all receive, a 2D image with the maximum amplitude $A_{max}(e_{ij})$ is obtained for each sequence, as in Fig. I.7(c). Similarly, a C-scan can be obtained by the displacement of a probe in a plane and by recording the maximum reception signal to create a 2D image.

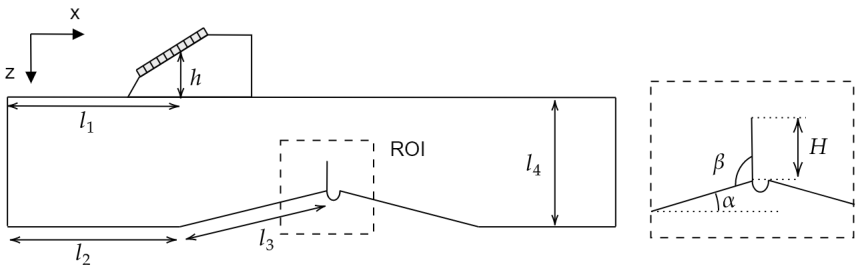


Figure I.8. Contact inspection configuration with a Rexolite wedge for imaging a back-wall breaking notch machined in a steel mock-up representative of a butt weld. Region of interest schematic representation on (x, z) plane to compute I^p. Some parameters of the setup are shown, such as specimen geometry (l_1, l_2, l_3, l_4, α), and flaw geometry definition (β, H), among others.

4. S-scan provides images of the ROI similarly to C-scan but is produced when a PA probe electronically sweeps an ultrasonic beam through a range of angles. In this case, the probe is fixed in position, so it is useful when the piece shape prevents continuous displacements in the inspection (e.g., a change of diametrical section in an axe). An image with axis $A(x, z)$ and amplitude (A) for each point is obtained.

The B-Scan, C-Scan, and S-Scan are imaging techniques in which echoes are computed and decomposed as various contributions or modes (L-T) in a multi-array transducer (PA). Modes correspond to various beam interactions with a scatterer (e.g., flaw or specimen surface) and to associated ultrasonic wave paths, including or not a reflection on another scatterer and possible mode conversion of longitudinal into transverse waves or vice-versa in the interaction.

A more advanced imaging technique called Total Focus Method (TFM) relies on Full Matrix Capture (FMC) acquisition technique. FMC uses a set of time-domain data (A-scans) from all combinations of transmitting and receiving elements on the transducer to generate 2D images of the ROI and its surroundings [39, 40]. The full matrix of array data used in the post-processing algorithm is called the Total Focus Method (TFM). In this algorithm, the beam is focused on a target region. The region is discretized into a grid represented in a (x, z) plane. The intensity of the image I^p on the discretized ROI is given by Eq. I.9. An example of a TFM image in a ROI is shown in Fig I.7(d).

Total focus method image is produced from a "delay-and-sum" algorithm applied to the set of N^2 inter-element signals obtained from the FMC acquisition schema with an array of N elements. If $s_{nm}(t)$ is the signal received by element m when element n is used as a transmitter, the image amplitude at a given point located by $\mathbf{r}$ consists in a coherent sum of N^2 analytic signals $\hat{s}_{nm}(t) = s_{nm}(t) + iH\{s_{nm}(t)\}$ at appropriate propagation times $t = \tau^p_{nm}(\mathbf{r})$ where H denotes the Hilbert transform. $\tau^p_{nm}(\mathbf{r})$ is the theoretical time of flight between transmitter n and receiver m through the image point at $\mathbf{r}$ for the p^{th} reconstruction mode, i.e., for one of the many potential ultrasound paths that can be exploited to form relevant images of a given crack-like defect. With these notations, the image amplitude $I^p(\mathbf{r})$ (example in Fig.I.7(d)) can be calculated as

$$I^p(\mathbf{r}) = \left| \sum_{n=1}^{N} \sum_{m=1}^{N} \hat{s}_{nm}\left(\tau^p_{nm}(\mathbf{r})\right) \right|. \tag{I.9}$$

I.5 . Multi-fidelity data on NDT&E

Regardless of the application field considered, numerical models can be categorized accordingly to their accuracy in simulating a given problem. Such a kind of differentiation is often referred to as the *fidelity of the model*. That is, when contrasting two models, they may be referred to as high- or low-fidelity models: the most accurate model is the high-fidelity (HF) model, while the less accurate one is the low-fidelity (LF) model. This means that fidelity is always defined by a comparison. Such a categorization applies to NDT&E and SHM simulations, too.

Data are also categorized by *fidelity levels*. One can consider the highest possible fidelity data to the measurements experimental setup that is completely mastered; thus, the impact of uncertainties is reduced to the minimum. Other source of high-fidelity may be in-situ recorded data. In the other hand, data coming from simulation may considered LF data. Simulated data are a fair representation of experimental or in-situ data but with some discrepancies. The source of those discrepancies are often uncertainties or non-considered phenomena in the LF model.

HF data has an intrinsic cost associated with its production. An HF model deployment or a real-world data acquisition represents a time-demanding procedure. This makes HF data less accessible than LF data in terms of quantity.

The multi-fidelity data term is intended to define, for a given problem, a setup (i.e., numeric simulations and/or experimental measurement) for which different data are exploited jointly. In the case of NDT&E and SHM inspection problems, relying on both experimental data and simulated data.

Apart from production costs, it's worth emphasizing that HF data often surpasses LF data in terms of quality, adding another dimension to the challenge beyond economic considerations.

Examples of different fidelity data are simulations from CIVA and experimental data (Fig. I.9). Even though the salient and meaningful image contents can be obtained through simulations (e.g., the defects echoes or reflections from ultrasonic waves), some signal perturbations may exist in experimental acquisition. Those perturbations are due to electronic noise or inspection conditions such as coupling between probe and specimen, among others. The perturbations generate patterns in experimental acquisitions that cannot be reproduced accurately in simulations.

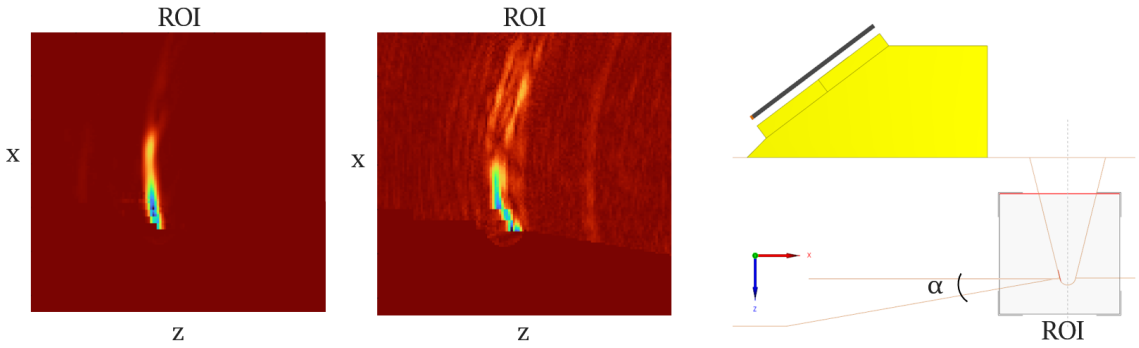


Figure I.9. UT-TFM image reconstruction of ROI in a welding joint inspection with a flaw by UT. Right: simulation output from CIVA. Middle: experience with similar flaw parameters and the same probe. Left: experience configuration with defined ROI. The differences between equivalent realizations on simulation and experimental acquisition can be appreciated, particularly in the echo signature.

CIVA is an example of a parametrical simulation model, where a large amount of data can be produced in a short time. Even if some factors in the field are not modeled, these models are suitable to represent a fairly set setup. However, some uncertainties are typically simplified or hard to model on forward simulation:

1. The human factor: the technique configuration may be set before the examination. However,

inspection can inject variability and noise into measures. Misplacing the probe or shifting the path direction can add a measurement error in the resulting data.

2. Inspection characteristics: Medium temperature and coupling compatibility can vary in the same specimen. Measurements in the identical specimens vary in each inspection campaign. These effects may be attributed to surface temperature but also operational setting alterations. As an example, imaging techniques reconstruction are affected by the incorrect parameter setting of speed sound.

3. Equipment characteristics: adaptation to the environment of acquisition equipment and sensors plays a role in experimental results not being considered in the modeling. Electronic noise derived from components, aging in electronics, or probes may introduce a bias in the inspection results.

4. Structural noise and attenuation: an equivalent scattering approach for an-isotropic properties in material simplifies the effect of grains and interface interaction (e.g., weld area properties represent uncertainty regarding added material properties). Experimental examples show notable differences when obtaining an image as output. The wrong assumptions about the internal structure of the specimen can lead to erroneous technique selection and unjust diagnostics.

5. Other sources of noise (environment).

Regarding machine learning, the HF data has less annotation or details of how it was produced compared to LF data. This characteristic of HF data presents an issue for machine learning and any application one wants to give to these data. However, HF is still significant since it contains valuable information about the real world, and in some cases, these data can be fundamental to deploying some applications on NDT&E, such as optimization, inference, uncertainty quantification, or statistical studies. As an answer to this problem, there is the possibility to develop multi-fidelity methods to take advantage of HF and LF data on NDT&E procedures. In [41], the authors categorize multi-fidelity methods according to three classes of strategies: adaptation, fusion, and filtering.

More into detail, adaptation proposes enhancing the low-fidelity model with information from the high-fidelity model while the computation proceeds. The fusion strategy is based on information fusion by evaluating low and high-fidelity models and combining information from all outputs. The filtering strategy invokes the high-fidelity model following the evaluation of a low-fidelity filter. Filtering strategy might entail evaluating the high-fidelity model only if the low-fidelity model is deemed inaccurate, or it might entail evaluating the high-fidelity model only if the candidate point meets some criterion based on the low-fidelity evaluation. In the context of this work, adaptation and fusion methods are preferred for the machine learning solution proposed later in the methodology. Filtering methods imply extensive access to the HF model (or acquisition procedure), which is very rare for NDT&E HF data.

In the context of this work, any solver shall be considered a function $\mathcal{F}$ that maps parameters or factors to a measurement space. For example, the mapping of $\mathcal{F}$ leads to the quantities $A(x, y)$ (Eqs. in Section I.4) for UT techniques or ΔZ for ECT techniques (Eqs. in Section I.3). $\mathcal{F}$ is the physic model and the ground truth to know how the parameters of a NDT&E procedure are related to the measurements. An image, a time signal, or even a vector of values can be considered in terms of measurement. If we call $\mathbf{p} \in \mathcal{P}$ the set of parameters that defines an inspection, and $\mathbf{Y} \in \mathcal{Y}$ a measurement of this procedure, in general terms, we can define $\mathcal{F}$ as a function that maps $\mathcal{P} \rightarrow \mathcal{Y}$,

$$\mathbf{Y} = \mathcal{F}(\mathbf{p}), \tag{I.10}$$

In the following sections, machine learning is introduced as a second pillar of this thesis work before introducing the methodology. One of the key ideas is to develop a machine learning surrogate model or metamodel ($\mathcal{M}$) that considers LF and HF data. $\mathcal{M}$ is intended to replace or complement $\mathcal{F}$ in NDT&E data production. $\mathcal{M}$ shall be a multi-fidelity model that provides a good representation of $\mathcal{F}$ while producing higher fidelity data compared to the fidelity data provided by $\mathcal{F}$.

I.6 . An introduction to machine learning: definitions, schemes and approaches

Machine Learning (ML) is the group of methods used to train a computer to automate a task. As a pioneer of artificial intelligence research, Tom Michell defined ML and the main parts of its deployment in the following sentence [42]:

Machine learning definition *A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.*

In the next sections, those elements (task, metric, and experience) will be recalled when describing particular algorithms. Tasks like regression or classification can be interpreted as what the program needs to learn. In the case of metrics, they are also named by 'cost' or 'loss' functions that quantify the error to minimize by a ML algorithm. Regarding the experience, it is the data or examples from which the algorithms learn. Other helpful definitions are shown in Table I.1.

Table I.1. Machine learning elements definition, based on [43] definitions.

Example, sample, or instance	Feature	Each example of an observed group can have several characteristics called features. For example, an image taken during a visual inspection presents its features by pixel values, while a recorded signal presents a feature value for each sampling through time.
	Label	Besides the features, an instance can be associated with a subgroup within the main observed group. The subgroup is commonly a set of parameters called labels. The label adds information about a class or a representative value of an example.
Input		Whenever the task is interpreted as classification or value estimation, the ML algorithm becomes a model that uses an example as its input to perform an estimation.
Data set or data		Assembly of examples or instances used as inputs. It also contains the labels (subgroups) assigned to the samples.
Output or inference value		Whatever the task performed by a ML algorithm, the result of evaluating the algorithm will be naturally a set of numerical values. These values estimate the computer program given to an input example. When the network is trained, this output is commonly an inference value.

Referring to Fig. I.10, ML approaches can be separated into Supervised Learning (SL), Unsupervised Learning (UL) and Semi-Supervised Learning (SSL) accordingly to the labels availability in the training phase. A fourth approach, reinforcement learning, is only mentioned since it is beyond the scope of the methods described here. That is, the supervised approach requires label data to train the model, while unsupervised training infers those labels (groups, categories, clusters, or classes in data) by itself. A third approach, semi-supervised, is practical when the data set labels are incomplete, or there is missing information in the data set that needs to be inferred by the algorithm. These approaches are recalled later during the methodology since all of them are explored to develop the presented framework. It is worth noticing that, in this work, the word *ML model* is used to refer to a functional (for instance, $f_{\boldsymbol{\theta}}$, with $\boldsymbol{\theta} \in \Theta$ a set of parameters to be optimized or trained) that takes numerical input values and delivers an output. ML model can have several forms but generically perform a task once trained. The next section describes the classification by performed task and model type.

I.6.1 . Supervised learning

The choice to use a supervised approach over the others is related to the task and the data set characteristics. The expected output from the ML model is known and used to train the algorithm. SL

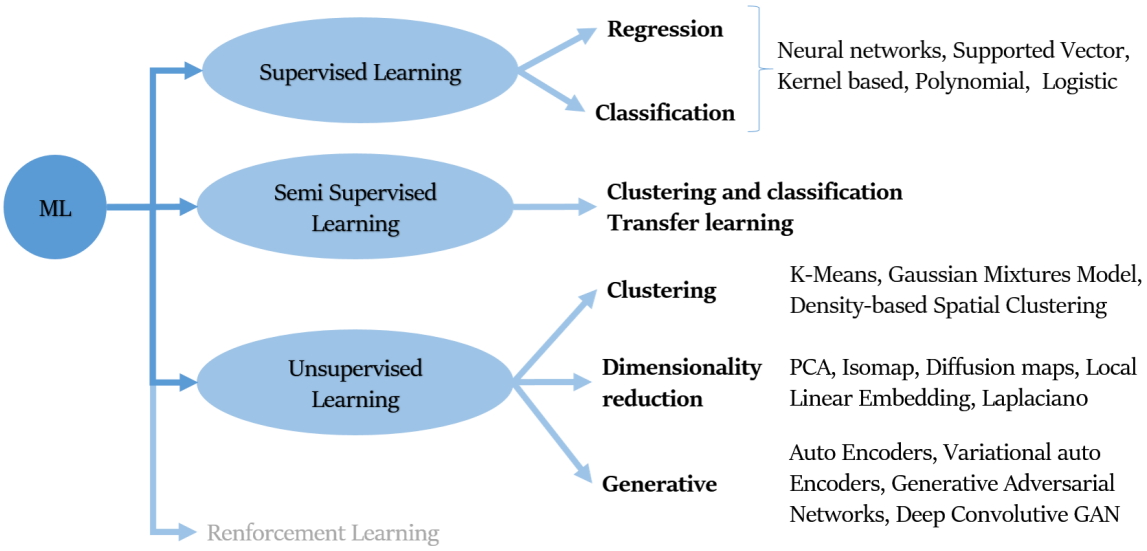


Figure I.10. Machine learning families and methods. Subcategories are described in the next sections. Based on [43] classification.

examples can be found in classification or regression tasks, which can infer a class or related labels for a given input.

The data set used for these tasks has observed features or inputs $\mathbf{x} \in \mathcal{X}$ being $\mathcal{X} \subseteq \mathbb{R}^D$ and targets $\mathbf{y} \in \mathcal{Y}$ being $\mathcal{Y} \subseteq \mathbb{R}^C$ for regression problems when predicting a set of scalars is required. If $\mathcal{Y} = \{1, ..., C\}$, then it is a classification problem where the class $\boldsymbol{y}$ is predicted for each sample.

The model learns to extract information from a training set $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n) : n \in \{1, ..., N\}\}$. For example, an instance $(\mathbf{x}_i, \mathbf{y}_i)$ can represent sensor values in time, voice recordings, coded words in a sentence, etc. on $\mathbf{x}_i$, which has a related class or a set of parameters $\mathbf{y}_i$, like the type of noise in the signal, the distance of the recorded source or style of writing, etc.

In a classification problem, the output vector $\hat{\mathbf{y}}_i$ is then the estimation of the classifier, expected to be the same discrete label $\mathbf{y}_i$ in the data set once the algorithm is optimized. The estimation error of the model can be evaluated by a function $\mathcal{L}(\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the model variables to be optimized. An application might be to predict material failure from data acquired by NDT&E techniques as in [44].

In contrast to classifiers, regression models estimate continuous outputs. In this case, taking $\mathbf{x}_i$ and computing $\hat{\mathbf{y}}_i$, the task is to infer a value, e.g., a material property from a grain structure image or the size of a defect given a set of record values in a sensor by optimizing the function $\mathcal{L}$, like in [4][45].

Data set partitioning: train, validation, and test Let us define $\boldsymbol{\theta}^*$ as the set of learn-able parameters of $f_{\boldsymbol{\theta}}$ that gives the minimum of $\mathcal{L}$ for a data set $\mathcal{D}$. $f_{\boldsymbol{\theta}^*}$ can be tested on a new set $\mathcal{T}$, with the same structure as $\mathcal{D}$ but $\mathcal{T} \cap \mathcal{D} = \emptyset$. The output of the trained model is used to compute any suitable metric that explains the performance of $f_{\boldsymbol{\theta}^*}$, given a task. The set $\mathcal{T}$ is the test set, a set of instances never seen during the training.

An additional set $\mathcal{S}$ may be used during the training, and this set is the $\mathcal{T}$ so $\mathcal{S} \cap \mathcal{T} \cap \mathcal{D} = \emptyset$. This set is called the validation set, used to compute the $\mathcal{L}_{val}$. $\mathcal{L}_{val}$ or validation loss is mathematically defined as $\mathcal{L}$ but computed with $\mathcal{S}$. This loss is not minimized but used as a generalization metric: if $\mathcal{L}_{val}$ decreases together with $\mathcal{L}$, $f_{\boldsymbol{\theta}}$ generalizes well.

Epoch definition A step during training of an ML algorithm in which a portion of examples in the training set $\mathcal{D}$ is used to infer output values. The outputs are used to calculate the error with the loss function for the portion of examples. The mean of the errors given by $\mathcal{L}_{\boldsymbol{\theta}}$ per instance is used to update weights, bias, and other trainable parameters in the NN. The back-propagation algorithm does the update

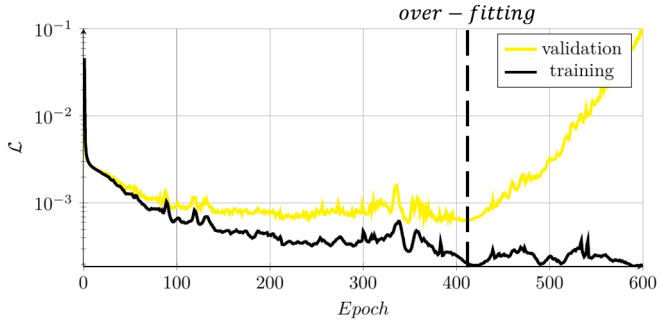


Figure I.11. Example of over-fitting in loss evolution. Validation loss $\mathcal{L}_{val}$ (in yellow) increases while training loss $\mathcal{L}$ (in black) decreases. $f_{\boldsymbol{\theta}^*}$ is founded after 400 epochs (dashed vertical line).

of $\boldsymbol{\theta}$. The portion of examples used for an epoch is named *batch*.

$\mathcal{L}_{val}$ may be used to avoid over-fitting during the training (Fig. I.11). A ML algorithm over-fits when it learns over $\mathcal{D}$, but it can not perform similarly over $\mathcal{S}$.
Similarly, $f_{\boldsymbol{\theta}}$ can under-fit if the trained was stopped before it learns all the possible from $\mathcal{D}$. This is generally corrected by changing $f_{\boldsymbol{\theta}}$ (e.g., increasing the number of parameters $\boldsymbol{\theta}$ to fit or training for more iterations). However, it is important to avoid $f_{\boldsymbol{\theta}}$ training too far or to have too many parameters since the contrary effect appears; this is over-fitting.

Looking out for $\mathcal{L}_{val}$ during the training to stop the iterations is called *early stopping*. This technique assures the best performance for a given $f_{\boldsymbol{\theta}}$ to generalize in the learned task when trained over $\mathcal{D}$ and checked over $\mathcal{S}$. $f_{\boldsymbol{\theta}^*}$ is tested later over $\mathcal{T}$ to quantify the generalization power during the test phase. The testing cannot be done over $\mathcal{S}$ or $\mathcal{D}$ since it shall introduce a bias during designing f.

I.6.2 . Unsupervised learning

Unsupervised learning (UL) can be separated by clustering, dimensionality reduction, and generative models (Fig I.10).

Clustering and dimensionality reduction methods use the set $\mathcal{X}$ which has no label $\mathbf{y}$. The main task of these methods is to infer new knowledge from data by sorting the examples by groups or categories that a priory is unknown. These methods are often named feature extraction. In this case, $\hat{\mathbf{y}}$ is unknown. Different criteria can be used to evaluate is $\mathcal{Y}$ is a fair parameter space for $\mathcal{X}$, commonly relying on probabilistic approaches. Identifying natural clusters of acoustic emission signals on NDT&E without previous knowledge about the data structure can be an example, as shown in [46].

UL methods vary from deterministic to stochastic models; data decomposition: Principal Component Analysis (PCA) [47], Independent Component Analysis (ICA) [48], singular spectrum analysis [49]; clustering in inferred groups; feature extraction: AutoEncoder (AE) and Variational AutoEncoder (VAE) and generative models. Some other methods are shared with SL, and in the same way, the methods above present a supervised variant. Supervised ICA [50] is an example, where $\mathbf{y}$ is the class being accessible to enhance the separability of features, and some hidden labels can be inferred better than in the case the class is unavailable. The last category in UL, generative models, is expanded in Sec. I.7.5.2 since it is extensively used for the aim of this work.

The rest of the UL methods commonly compare two or more different data sets in a reduced and more readable representation (e.g., high dimensional data into a 2D representation). Feature extraction methods help understand relations between a given sample and the complete data set: singularities or similarities with other samples. Unlabeled data can be categorized using these methods by ‘clusters’ to represent or understand these similarities.

I.6.3 . Semi-supervised learning

Semi-Supervised learning (SSL) combines clustering methods from unsupervised categories with supervised classification methods to fulfill a classification task when the data are not fully labeled in all the examples. These unbalanced labeled data sets are commonly found on NDT&E and SHM where bad labeling, and lack of knowledge about how data were produced, among others.

The possibility to cluster the data by unsupervised methods is adapted to any data set, even when it is not labeled. For example, if incomplete information about the classes exists, some instances are labeled, and the learned subgroups (clusters) can be associated with those classes. Therefore, the class can be inferred for any unlabeled data, and new classes that do not exist in the original labels can be inferred.

No specific ML methods are reserved for SL, US, or SSL. They are commonly combined with different optimization problems depending on the goal of the task.

In future sections, special attention will be given to representation and generative learning as an UL and SSL methods (see I.7.5 and I.7.5.2) due to their potential in achieving the objective in this thesis.

I.7 . Background on machine learning methods and techniques

ML takes part in a more prominent family named Artificial Intelligence (AI), described in the bibliography as a machine with cognitive abilities similar to human intelligence. An AI shall not have all the human capacities but can imitate some more accurately and efficiently [51]. From another perspective, AI explores the mathematics behind our possibility to perceive, think, and act [52].

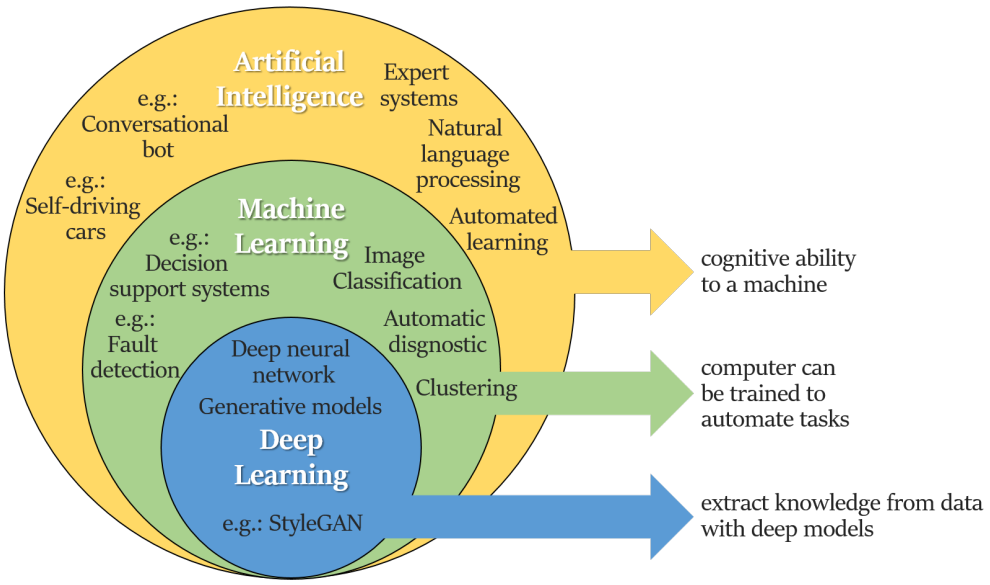


Figure I.12. Artificial intelligence subgroups. Some examples of methods are listed in each level for illustration. Based on [53] definitions.

Either a ML or a DL method aims to create a model that performs a specific task. A good performance may be achieved by training (or optimizing) the chosen algorithm in an iterative way. Different methods to train the algorithms are introduced in section I.7. From here, ML will be used to generalize ML and DL terms. A representation in Fig. I.12 shows the subgroups of AI, ML, and DL and the hierarchical link between them.

Before going into the ML techniques definitions, schemes, and approaches, it is essential to mention that ML methods[1] often employ Neural Networks (NN). A NN is a sequence of interconnected artificial

[1]In this work, the words 'approach', 'method' and 'technique' are used as synonyms. Nevertheless, 'method' or 'approach' are preferable since this introduction is intended to be a general overview of DL. Later, some so-called 'method' (or 'approach') here may be named as techniques since we are closer to

neurons interpreted as a set of functions that can be represented by directed acyclic graphs [54]. The nature of an artificial neuron can vary depending on the task, with the perceptron (described in Appendix VI.2) or convolutive neuron being the most common in the bibliography (described later). A neuron is then a sub-function in the NN that has parameters to be optimized.

This section is a non-extensive review of the methods in machine learning. Later, in Chapters II to IV, they will be recalled on specific study cases.

I.7.1 . Regression and classification methods

Different methods to create a model to perform a regression in ML exist. In Fig.I.13, two main categories present linear and non-linear methods. In the first category, the models are typically a polynomial function $\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$ with $\boldsymbol{\theta}$ being linear coefficients or hyper surfaces in n-dimensional problems, e.g., when input $\mathbf{x}$ has $N > 1$ components. Building f varies on every method. Regularization to train a model is described in the next sections but mentioned as a sub-method in regression (Ridge, LASSO, or Elastic Net). Non-linear methods introduced two remarkable methods to build a model: NN and kernel-based regression (Appendix VII).

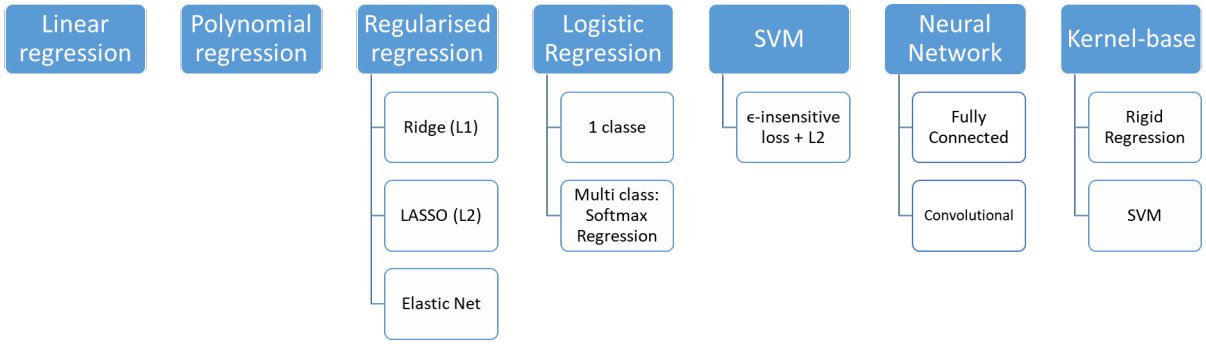


Figure I.13. Diagram representing an overview of the main regression methods [55][43][56].

The same techniques are also found in classification with some differences. For instance, the main difference is that either the polynomial model, the hyper-surface, or the non-linear method defines regions in a space to categorize each input, in other words, to give a class label. To clarify this idea, it is possible to imagine an n-degree polynomial model used as a regressor, where x and $\hat{y}$ are uni-dimensional continuous values. By comparison with a classification case, the model represents a bounder for two classes in a (x, y) plane (Fig I.14).

Feature extraction techniques (Section I.7.5) are frequently used as pre-processing before using a regressor or classifier to improve its performance. Deep learning uses an unique and more complex model that can perform feature extraction and classification (or regression) tasks. Compared with feature extraction, in the last case, a large amount of data is required to ensure that the model learns all the features in the data. It is also required that all classes or groups are sufficiently represented in the data set. No algorithm will learn what is not present in the input data, but it can learn to generalize fairly well to predict unseen samples when the seen data are representative enough. For instance, a neural network will not learn to identify a cat between a group of animals if an example of a cat is not presented to the network during the optimization.

Regarding NN, Fully Connected Neural Networks (FC-NN) and Convolutional Neural Networks (CNN) are used in regression and classification. They are described in more detail in Sec. I.7.2 and Sec. I.7.4 in general since they are used not exclusively in regression and classification but also in data decomposition, data projection or deep generative models.

---

a specific implementation, so more details are given.

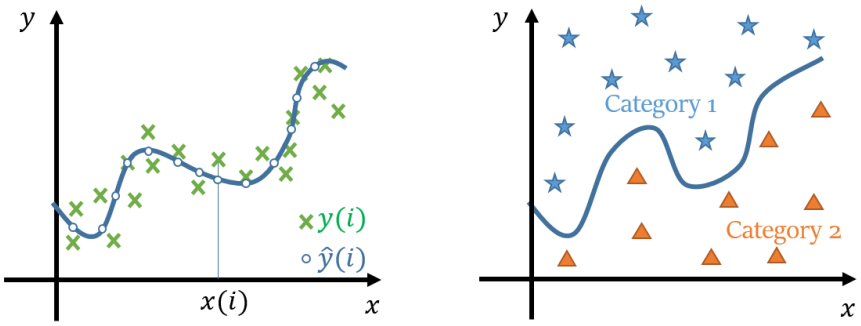


Figure I.14. Regression and classification: Similar polynomial models are used in a regression problem (right) and a classification problem (left). The figure represents a 1D input (x) and 1D output (y) model. The blue line is $f_{\boldsymbol{\theta}}$; at the right, $f_{\boldsymbol{\theta}}$ gives a value for every input; at left, it represents a boundary in the (x, y) plane to separate classes, being x and y a 2D input and Category 1 and Category 2 the output for $f_{\boldsymbol{\theta}}(x)$.

I.7.2 . Machine learning approaches based on deep artificial neural networks

This section introduces some concepts for deep artificial neural networks. Some definitions can be found on Appendix VI.2.

Deep learning algorithms differentiate themselves compared to the other ML methods since they rely on hierarchical architectures aiming at extracting suitable information from the input features based on the use of non-linear affine transformations performed on the data. The generalization of NNs to architectures having more than one layer is referred as Deep Neural Network (DNN). More data and computing power are needed when applied due to the nature of the algorithms: a large amount of parameters to optimize.

Many samples of deep models exist to infer information from a data set. Fig. I.15 presents a typical dense layer architecture, an example base architecture to build Deep Neural Networks (DNN). This kind of neural network is described in detail in Appendix VI.2. Complementary concepts that are recalled in this section and later, such as the perceptron unit, loss functions, hyper-parameters in NNs, and back-propagation algorithm, are explained in more detail in the Appendix VI.2.

Normalization on NN, convolutional neural networks, among others concepts are introduced here because of the relevance for the methodology in the next Chapters.

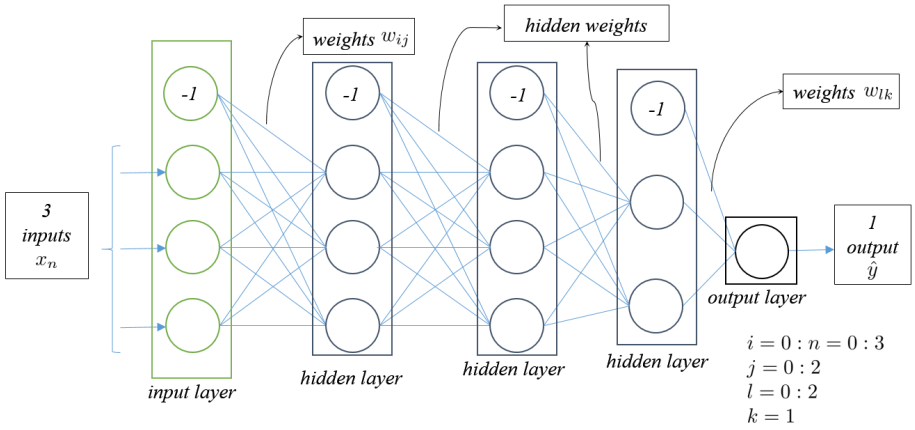


Figure I.15. Fully connected network or multi-layer perceptron architecture with one hidden, three neurons in the input layer, and one neuron in the output layer.

I.7.3 . Normalization, regularization, and drop-out layers

Normalization, regularization, and dropout are essential techniques in machine learning, especially for training DNNs. They help stabilize training, prevent over-fitting, and improve the model's generalization ability to unseen data.

Before talking about normalization and drop-out layers in NN, the inner features concept (also called extracted features or activation values) must be introduced. In a feed-forward NN, each neuron layer can be seen as a functional $\mathscr{L}_l$ mapping the inputs into the outputs, so $\mathbf{x}_{f_{l+1}} = \sigma \circ \mathscr{L}_l(\mathbf{x}_{f_l})$, where l is the number of considered layers, $\mathbf{x}_{f_l}$ is the activation values from the precedent layer, and $\mathbf{x}_{f_{l+1}}$ is the resulting activation values after passing $\mathbf{x}_{f_l}$ through $\mathscr{L}_l$ and σ. Like an instance in a data set, $\mathbf{x}_{f_l}$ has a defined dimension. The dimension of the features depends on the layer configuration (e.g., number of neurons). In FC-NN, due to the application, vectorization (or flattering) and non-linear transformation between layers, the extracted features lose the meaning associated to pixels or amplitude values one finds in the data set used, i.e., an image or time signal sample $\mathbf{x}$, turning them into an abstract representation of an instance during the feed-forward operation of the NN.

Normalization is the pre-processing procedure aiming at scaling and shifting the input features before training. However, normalization can also be performed on inner features. Such a specific normalization is a ML technique with an additional normalization layer. An example of NN f_θ (Eq. VI.3) modified so every neuron layer implements a normalization layer $\mathscr{N}_l$ is

$$f_\theta(\mathbf{x}) = (\mathscr{N}_L \circ \mathscr{L}_L \circ \sigma_L \circ \mathscr{N}_{l-1} \circ \mathscr{L}_{l-1} \circ \sigma_{l-1} \circ ... \mathscr{N}_1 \circ \mathscr{L}_1 \circ \sigma_1)(\mathbf{x}), \tag{I.11}$$

$$\text{with} \begin{cases} \theta : \text{set of weights and bias} \\ l = 1 : L \\ L : \text{number of layers on } f \end{cases},$$

$\mathscr{N}_l$ can normalize $\mathbf{x}_{f_{l+1}}$ in different ways. Batch Normalization (BN) [57], Instance Normalization (IN) [58], and Layer Normalization (LN) [59] are examples of different normalization layers. Each of them computes a shift β and scale γ to apply from the activation values as follows

$$\mathscr{N}_l(\mathbf{x}_{f_{l+1}}) = \gamma \cdot \frac{\mathbf{x}_{f_{l+1}} - \mathbb{E}[\mathbf{x}_{\mathbf{f_{l+1}}}]}{\sqrt{\sigma^2[\mathbf{x}_{\mathbf{f_{l+1}}}] + \epsilon_n}} + \beta, \tag{I.12}$$

where $\mathbb{E}[\cdot]$ and $\sigma^2[\cdot]$ represent the empirical average and variance, respectively. ϵ_n is a small value for numerical stability. The main difference between BN, IN, and LN is how $\mathbb{E}[\cdot]$ and $\sigma^2[\cdot]$ are computed. For example, BN computes $\mathbb{E}[\cdot]$ by batches. It normalizes the activation of a layer by calculating the mean and variance of the activation values across the entire batch for each feature. Conversely, IN operates at the level of individual instances (i.e., samples) within a batch. It normalizes the activation of each instance separately.

Consequently, with the $\mathbb{E}[\cdot]$ and $\sigma^2[\cdot]$ computation, an optional scale γ and shift β can be added to the normalization operation. β and γ are the learned parameters of f_θ. The objective of this is to introduce flexibility into the normalization. β allows for shifting the normalized values to a learned offset, which can help handle situations where the mean-shifted values are not optimal at zero. The γ parameter enables the normalized values scaling, allowing the network to control the magnitude of feature representations.

In deep learning, with regularization, one intends a set of techniques used to prevent over-fitting. Over-fitting occurs when a model learns to fit the training data too closely and fails to generalize to new, unseen data. Regularization methods add a penalty term to the loss function, discouraging the model from learning overly complex patterns. In the same way, drop-out is used as a regularization technique. Drop-out is added to the architecture as a layer (like for normalization). This layer randomly sets a fraction of the neuron outputs of a layer to zero with each forward pass.

I.7.4 . Convolutional neural networks

An architecture that competes with FC-NN is introduced in this section. Convolutional Neural Networks (CNN) are extensively used in NDT&E classification and regression problems [60][61][62][44][45] since they can find defects in NDT&E by 2D images. Unlike neural layers on FC-NN, CNN architecture is based on *convolutional layers*. These layers can represent the hierarchy of the elements on images by learning spatial or temporal correlated features. In this case, the layer mimics how the human visual cortex works [63]. Mimicking is done by keeping the pixel relation through the network: convolutional layers create feature maps (also known as activation values) that are spatially correlated. In contrast, in FC-NNs, the spatial correlation is lost from the beginning of the forward propagation by flattening the input.

The CNNs are built by the convolution, activation, and pooling layers, often referred as convolutional blocks. The activation layer is like on FC-NNs, so it is not trainable. The pooling layer keeps no parameter to train either, so the only layer that participates in the back-propagation algorithm is the convolutional layer: it contains weights.

Convolutional layer This layer is defined by a weighted kernel represented on a matrix. The kernel operates on the input to produce a feature map. A layer may contain many kernels in order to generate many feature maps. The operation between the input and a kernel is a convolution operation. An example is shown in Fig.I.16. The convolution has as hyper-parameters the padding and the stride: columns and rows of values added to the limit of the image with specific values (typically zero) and number of pixels to move the kernel over the image in each convolution step. The kernel values (also called weights) are the learn-able parameters for this layer, meaning that they will be optimized similarly to a FC-NN by the back-propagation algorithm.

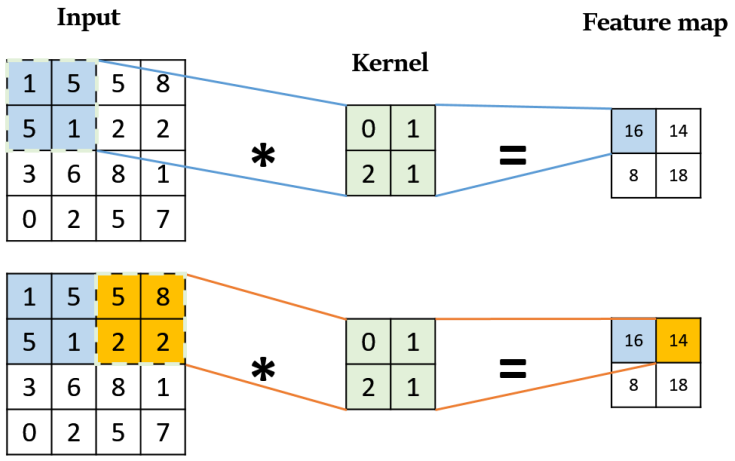


Figure I.16. $2D$ convolution operation. The stride is set to 2 in both directions, with no padding. The first operation (above) shows how the convolution starts with the first region in blue to get the first element at position $(0, 0)$ of the resulting feature map. The second operation (below) applies the convolution to a second region (in yellow) of the input to get the second element at position $(0, 1)$ of the resulting feature map.

The example case shows a single-channel $2D$ kernel with a single-channel input. An input can contain many stacked $2D$ maps*;* each $2D$ map is a *channel*. For instance, colorful images represented by a numerical input have more than one channel (e.g., three-channel for RGB images). Consequently, a kernel can also have more than a single-channel (3 channels for RGB images). Regarding the $2D$ feature map channels, it depends on the number of multi-channel kernels included in the operation; for each multi-channel kernel, a 2D activation map is obtained after the convolution.

A practical example is an RGB image represented by (4 pixels, 4 pixels, 3 channels) convoluted by a kernel of size $(2, 2, 3)$ with 12 weights. The resulting $2D$ convolution operation output is a (2 pixels, 2

pixels, 1 channel) feature map. If an additional kernel of size $(2, 2, 3)$ is added to the operation, 12 more parameters are added, and the output turns to have a size of (2 pixels, 2 pixels, 2 channels).

Pooling layer The pooling layer acts after the convolution layer and activation function. This layer reduces the spatial size of the feature maps by a dimensional reduction. It aims to extract dominant features and mitigate the sensibility to feature location: the classifier or regressor needs only to compute the value or class, not to keep its location.

Pooling layers may be Average-pool or Max-pool type, and the size determines how much the information from the previous layer is condensed. The stride and pooling are also hyper-parameters in this layer, like for the kernel in a convolutional layer.

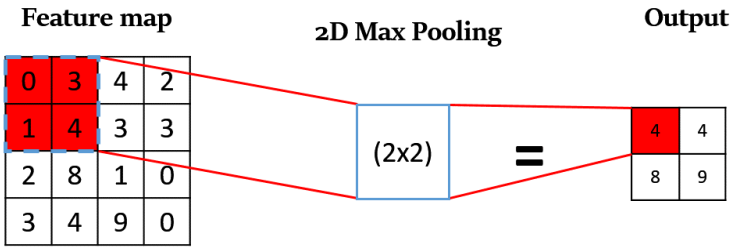


Figure I.17. Max pooling operation. The stride is set to 2 in both directions, and there is no padding.

This CNN architecture is faster on training convergence because it reduces parameters compared to a FC-NN for the same task. An example of it is when the input size is $D = (28, 28, 1)$, the first layer on a conventional FC-NN with 10 neurons sums up more than $8\,k$ connections and trainable weights. In the simple CNN case, the convolutional layer keeps a fixed amount of connections even on a more considerable input. The amount of trainable weights is linked to the kernel hyper-parameters: the same input can be processed by 6 kernels of the size $(2, 2)$ as in Fig.I.16, giving a total of 40 trainable weights. The rest of the layers (pooling and activation) are non-trainable. The resulting convolution operation produces a reduced feature map of size $(7, 7, 6)$. A final 10-neuron dense layer after flattening the reduced feature map adds to this CNN $\sim$ 3 k trainable weights to the architecture but competes in performance and training time with the FC-NN of 8 k weights. A more in-depth analysis of performance in real scenarios can be found in the bibliography [64, 65].

As mentioned, many architectures on CNNs implement a dense layer at the end of the convolutional layers to extract information from the last feature maps and generate the output. This last layer needs significantly fewer parameters than in the case of an architecture only with fully connected layers.

The CNNs can be represented by the Eq. VI.3. The main difference is the layer Eq. VI.4. The equivalent for Eq. VI.4 for a convolutional layer in Fig.I.16 is,

$$\ell_{ij}^{k}(\mathbf{X}^{k-1}) = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\left(\omega_{ab}^{k} x_{(i+a)(j+b)}^{k-1} + b^{k}\right) \quad \text{with} \quad \begin{cases} \omega : \text{weights for layer } k \\ b : \text{bias for layer } k \\ m : \text{2D kernel size} \end{cases}, \tag{I.13}$$

where ℓ_{ij}^{k} represents the ij-activation of a neuron in layer k, being ℓ^{k} a $2D$ output matrix. The kernel of size $(m, m, 1)$ has weights denoted by ω_{ab}^{k}. ℓ_{ij}^{k} is calculated for an $2D$ input matrix $\mathbf{X}^{k-1}$ has $x_{(i+a)(j+b)}^{k-1}$ elements.

The back-propagation algorithm is applied to the CNN layer like in the dense layer, so the equations do not change for the optimization process in convolutional layers, even if different layers take part of the architecture (e.g., CNN and FC-NN architecture). Normalization, drop-out, and regularization concepts from Subsection I.7.2 are also applied in CNN architectures.

I.7.5 . An overview of dimensionality reduction techniques: data decomposition, manifolds and data projection

In this section, we shall introduce some notions on the most used algorithms as applied to feature extraction and data projection/visualization. A common ML practice is analyzing data in a lower-dimensional representation. This representation is often known as a manifold. Regression and classification problems can be affordable when a representative data set has enough samples of the classes or parameter space. Even when these two characteristics are fulfilled, a too-big dimension D of $\mathcal{X} = \mathbb{R}^D$ (e.g., high-definition images or large parameter vectors) may represent a problem due to sparse data [66, 67].

For example, UT data can produce high dimensionality data for, e.g., a time signal with a small sample time or when it represents an image with three color channels. A recorded pulse-echo amplitude vs. time signal at $200\,\mathrm{MHz}$ from UT inspection (A-Scan) during $5\,\mu\mathrm{s}$ has 1024 sampling points (or features). Another example of high dimensionality is an image of a ROI obtained by B-Scan in UT. The case of a $(200, 200)$ pixels image with three color channels reaches a total of $120\,000$ features.

Precedent cases represent a high-dimensional problem for the techniques presented in Sec. I.7.2 since the sample needs to be flattened (in the case of an image) before forwarding it into the input. High-dimensional features drive the FC-NN to increase the number of parameters to train in the model and, consequently, the training time. Techniques in Sec. I.7.4, where no flattening is required, may decrease the training time for high dimensional spatial or temporal data (recurrent data type in NDT&E field).

Another issue with high dimensional data sets is the data sparsity, which can drive to missing classes or a low representation of one or more particular classes on the data set. Compared to small-dimensional data, high-dimensional data tends to need large instances for each class. Sparsity represents an issue even for techniques in Sec. I.7.4: a high dimensional data set constitutes a risk of over fitting the model only to classes that are represented enough, leaving the rest out of model generalization. This predicament is titled on the bibliography as the 'curse of dimensionality'.

This section presents the approaches to work around this common problem by reducing the data dimensionality. Data decomposition, manifolds, and data projection techniques applied in dimensionality reduction are named and classified.

Given a mapping function $F : \mathbb{R}^D \to \mathbb{R}^d$, where $D >> d$, F shall be capable of representing the original data set in a lower dimension without losing information. Principal techniques to find F are listed in Fig.I.18 and Fig.I.19.

The techniques can be presented as convex or non-convex optimization problems [68]. In the first group, all constraints of F are convex functions, and the solution is a global optimum. These techniques can be separated by full spectral techniques that are an eigen-analysis of a matrix built from the data set (e.g., Principal Component Analysis (PCA) or Independent Component Analysis (ICA)). In the case of sparse data, some particular solutions like Locally Linear Embedding (LLE) and Laplacian Eigenmaps are proposed in the bibliography as manifold techniques.

Sometimes, convex techniques are not sufficient to simplify its representation. Non-linear relations between input features in data often lead to kernel-based techniques like kernel PCA. In these cases, the input of the functions is a feature space constructed by employing a kernel function (See appendix VI.1 for kernel definition and appendix VI.4 for PCA definition). All convex techniques are enumerated in Fig. I.18 under categories.

In the case of non-convex techniques, they have objective functions to be optimized with many local optima, and its result varies in each execution, even in the same data set, due to the nature of the optimization method. t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) are widely used in the bibliography as dimensionality reduction techniques to enhance various ML methods. However, they are more extensively used as data visualization techniques [69].

Not all these data decomposition, manifolds, and data projection techniques are developed in this introduction. Only the relevant details for the later described methodology are exposed from here.

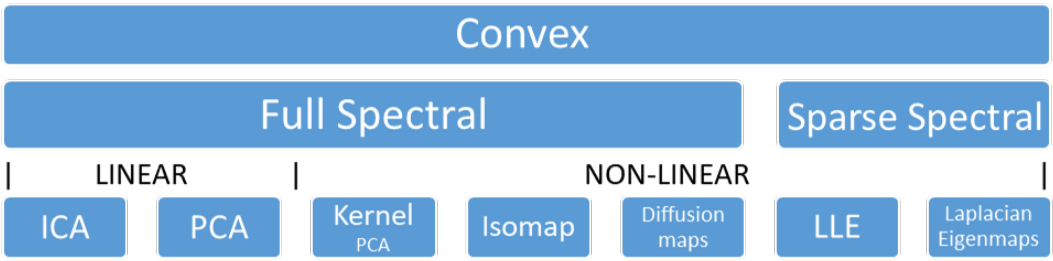


Figure I.18. Dimensionality reduction convex techniques grouped by full and sparse spectral application.

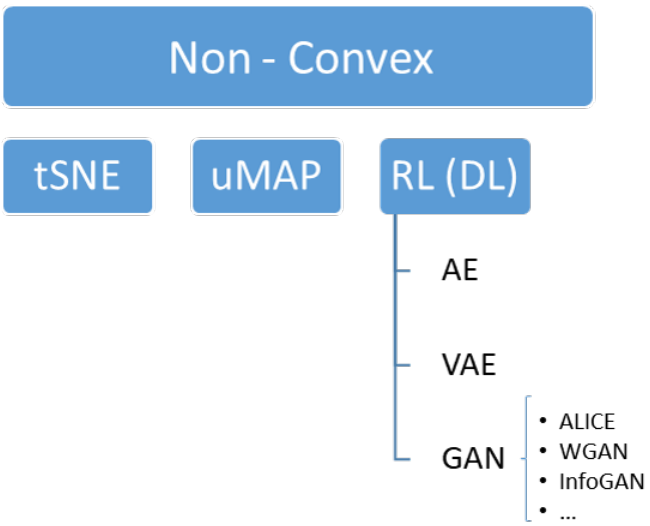


Figure I.19. Dimensionality reduction non-convex principal techniques. RL stands for representation learning, and DL stands for Deep Learning. AutoEncoders (AE), Varational AutoEncoders (VAE), and Generative Adversarial Networks (GAN) learn a representation of the data set based in DL architectures.

I.7.5.1 . t-SNE and UMAP projections as visualization tools

Data projection techniques can be helpful to visualize hidden data set proprieties such as clusters or sub-groups, to find relations between data such as similar samples in the data set, or to visualize better a high-dimensional data set. For this, two techniques are often used in ML: t-SNE and UMAP. Both projection techniques were conceived to generate a $2D$ or $3D$ visualization of a high-dimensional data set where the number of features is more than three.

t-distributed Stochastic Neighbor Embedding (t-SNE) [70] is based on distances between samples in the data set modeled by probability laws. It is possible to imagine that an instance x_i can be represented by a point on a high dimensional space, the data original space. The algorithm proposes to center a Gaussian distribution in each point of the original space. The distribution is built by using the distances in the original space. The conditional probability to an instance x_j near x_i is given by

$$p_{i|j} = \frac{e^{\frac{-||x_i - x_j||^2}{2\sigma_i^2}}}{\sum_{k \neq l} e^{\frac{-||x_k - x_l||^2}{2\sigma_i^2}}}, \tag{I.14}$$

where $||\cdot||$ is an Euclidean distance in the original space, σ_i is the standard deviation that depends on a hyper-parameter called perplexity, interpreted by the number of desired neighbors to be found around

$x_i.p_{i|j}$ can be interpreted by a way to measure similarity between neighbors, the closer regarding $||\cdot||$ they are in the original space, the higher is $p_{i|j}$. When $p_{i|j}$ is close to one, x_i and x_j are considered to be similar.

The term $e^{\frac{-||x_i-x_j||^2}{2\sigma_i^2}}$ can be seen as a projection of x_j on a 'distance axis' centered on x_i, where x_j has an assigned probability given by this term. This term must be regularized by $\sum_{k\neq l} e^{\frac{-||x_k-x_l||^2}{2\sigma_i^2}}$; otherwise, two $p_{i|j}$ and $p_{k|j}$ distributions would assign largely different probabilities to the first of their local neighbor. The Eq. I.14 is then normalized by the sum of the projection values for the rest of the samples in the data set ($k \neq j$). This provides $p_{i|j}$ a relative value regarding the the rest of distances and allows the algorithm to model the distances globally instead of locally, particularly when some clusters are found thanks to this projection.

The second part of t-SNE algorithm is to create a low-dimensional space to represent the points of the data set. New probability distributions are built for this reduced space. Each x_i has a coordinated correspondent y_i in this space. This time, a t-Student ($\nu = 1.0$) distribution is chosen to model similarities between points in this space since this distribution has a longer tail, the probability for short distances are not too close to 1.0, and further distances are not too close to 0.0. Since t-Student models the visualization space ($2D$ or $3D$), the distribution choice gives a less concentrated cluster of points. The conditional probability to an instance y_j near to y_i in a reduced space is given by

$$q_{i|j} = \frac{(1+||y_i-y_j||^2)^{-1}}{\sum_{k\neq l}(1+||y_k-y_l||^2)^{-1}}, \tag{I.15}$$

$q_{i|j}$ acts as $p_{i|j}$ but in the reduced space. Now that there are two conditional probability functions in each space, we can compare them. It is possible to use the Kullback-Leibler (KL) divergence between the two conditional probabilities as a comparison metric. A gradient descent takes y_i as a dependent variable. The objective is to find the set of coordinates y_i in the reduced space that minimizes KL-divergence. In other words, the similarities given by $p_{i|j}$ match those given by $q_{i|j}$.

An example of a t-SNE reduced representation on a 3D space of the MNIST data set [71] is shown in Fig. I.20, together with a PCA representation of the same data. MNIST dataset is a 2D array of pixels, where each element is a gray-scale value representing pixel intensity, and the image is 28 pixels in width and 28 pixels in height. Therefore, the dimensionality of each image in the MNIST dataset is 28x28 = 784. Here, the 784 features are reduced to 3 to represent clusters of the hand-written number data set.

Other representations compete with t-SNE, like UMAP [72]. This visualization is less applied and explored since it is a newer data projector. However, the nature of the algorithm gives a deterministic result, which is an attractive characteristic for repeatability and comparison.

I.7.5.2 . Representation learning and generative learning

Representation learning (RL) implements NNs to extract meaningful patterns from data to create representations for $\mathcal{D}$ through a function F. Unlike PCA or UMAP, F function relies on a set of NNs. Both FC-NN and CNN play an important role in the state-of-art of RL techniques. In this approach, F is an arrangement of FC-NN and CNN trained over the data set. F is often called filter or encoder. They are commonly applied in a DL schema since they use several layers, or even several NNs. Here, dimensionality reduction is not the only objective. The set of NNs may be deployed later as new data generators.

Most of the architectures for RL are also *generative networks.* This means that besides F, there is also a G function (called generator or decoder) that can generate never-seen data. In other words, it can increase the known data set $\mathcal{D}$ with new instances. f_θ is now separated into two NNs, and equivalent to $F \circ G$. This type of RL architecture is focused here since it is implemented later in this work. Representation and generative learning are strongly dependent, so both concepts must be developed together.

F and G functions can take several forms depending on the architecture and the task. In any case, F is intended to create a new reduced representation of the data, while G is intended to use the learned reduced representation to create new data.

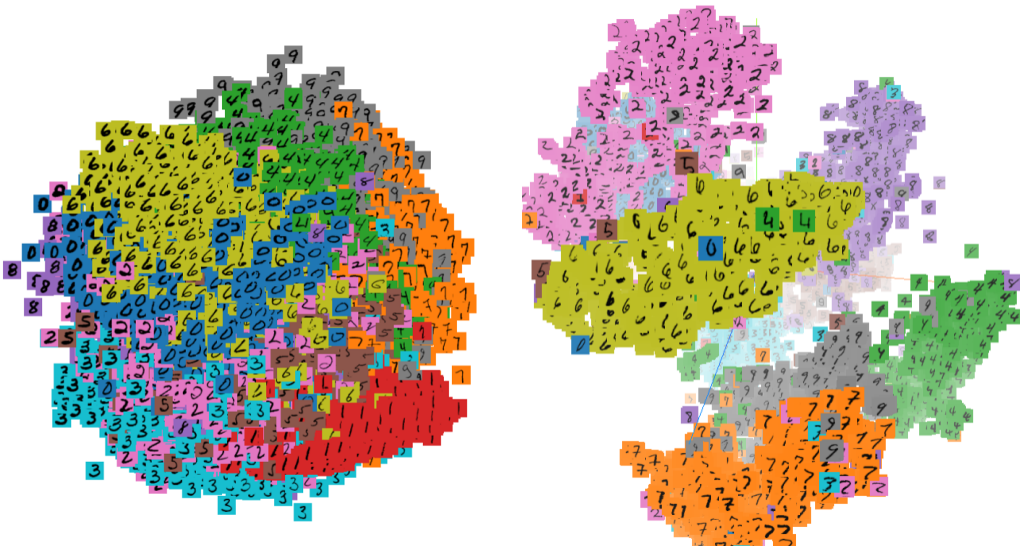


Figure I.20. PCA (left) and t-SNE (right) representation in 3D of the MNIST data set. In PCA representation, each point is represented by the coordinates given by the first 3 principal components. In t-SNE, by a 3D $\boldsymbol{y}_i$ vector. Each point is plotted in a different colored square with the corresponding hand-written number for both representations. The t-SNE presents a better cluster separation than PCA.

Some of the most primitive RL architectures exposed in this subsection are AutoEncoders (AE) and Variatonal AutoEncoders (VAE). A separate subsection is given to Generative Adversarial Networks (GAN). They are differentiated from AE and VAE architectures because GANs implement adversarial training, described later in Section I.7.6. Besides this main difference, GANs have also shown an impressive capacity for data generation by learning a meaningful representation of the data set, so they deserve special attention for the objectives of this work.

AutoEncoders It was mentioned before that some UL techniques relied on NNs architectures. AEs [73] are an example where the task is to learn a reduced representation of the data set while following a reconstruction objective. Each layer in a network F produced an output that can be interpreted as a new representation of the original input instance $\mathbf{x}$. If the last layer of the network F contains smaller quantities of neurons than the original dimension ($d << D$), the outputs of F are interpreted as a reduced representation. The representation is learned by the weights of G and the reconstruction objective: G takes the outputs of F during the training to reconstruct the $\mathbf{x}$.

In order to train both networks, the architecture is measured on performance by the ability of G to reconstruct the data after passing through the F bottleneck (reduced dimension). Different metrics can be proposed regarding the reconstruction task. L-2 norm in Eq. I.16 is an example of loss function $\mathcal{L}$ for the reconstruction objective over the N instances in a data set $\mathcal{D}$.

$$\mathcal{L}_{AE} = \sum_{n=0}^{N} \|\mathbf{x}_n - G \circ F(\mathbf{x}_n)\|_2^2 \tag{I.16}$$

An example of a simplified AE architecture is shown in Fig. I.21. The output of the first part of the AE (F) is named *latent space* or *coding*. Like other dimensionality reduction techniques, an AE learns to reduce $\mathbf{x}_n$ into a representation $\mathbf{z} \in \mathcal{Z}$. F is a mapping function $F : \mathcal{X} \to \mathcal{Z}$ trained to create a latent space $\mathcal{Z}$ from $\mathcal{X}$. G is another mapping function responsible for reconstructing data during training. The generator or decoder $G : \mathcal{Z} \to \mathcal{X}$ map the latent representation $\mathcal{Z}$ to data space $\mathcal{X}$. $\mathcal{Z}$ is a learned reduced representation of $\mathcal{X}$ where $F(\mathbf{x}_n) = \mathbf{z}_n$ and $\mathcal{Z} \subseteq \mathbb{R}^d$, being $d << D$. To simplify the notation, we will call the reconstructed sample from G as$\hat{\mathbf{x}}_n$, obtained from $\hat{\mathbf{x}}_n = G \circ F(\mathbf{x}_n)$.

Unfortunately, once AE is trained, the latent space structure is unknown, hindering the use of AE as

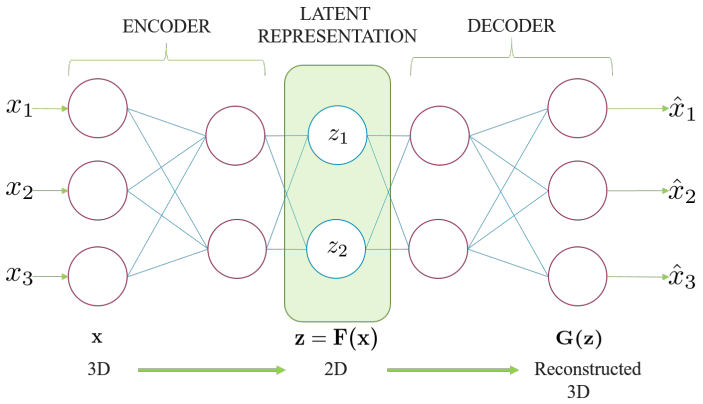


Figure I.21. Auto Encoder architecture using a FC-NN network with a coding representation and reconstruction network. Latent space $\mathcal{Z}$ is a reduced one dimension representation of the original data set.

a data generator. The more straightforward way would be to use the trained G to generate new instances $\mathbf{x}_{gen} = G(\mathbf{z}')$, where $\mathbf{z}' \in \mathcal{Z}$ but does not have its origin on $F(\mathbf{x}_n)$ from $\mathcal{D}$. VAE techniques proved to be more suitable for using G as a generator to create new never seen data by the NN in a proper way. This is linked to the fact that we have more information about $\mathcal{Z}$ structure than in AEs.

Varational Autoencoders Variational AutoEncoders [74] represent and generate data relying on a statistical approach. The learned latent space is not a numerical representation (vector) but a learned normal distribution for each z_i. VAEs induce the latent space distribution, for instance, to a normal distribution during training by using Kullback–Leibler (KL) divergence as a metric. The loss function in this case is described in Eq. I.17. Compared to $\mathcal{L}_{AE}$ from AE, $\mathcal{L}_{VAE}$ KL-distance term acts as a regularizing term, so $\mathcal{Z}$ distribution is known at the end of the training.

$$\mathcal{L}_{VAE} = \sum_{n=0}^{N} \|\boldsymbol{x}_n - G \circ F(\boldsymbol{x}_n)\|_2^2 + \sum_{n=0}^{N} D_{KL}(\mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n^2) || \mathcal{N}(0, 1)) \tag{I.17}$$

In practice, F outputs consist on two vectors, denoted as $\boldsymbol{\mu}_n$ and $\boldsymbol{\sigma}_n$, where component i represents the parameters of a distribution of z_i in $\mathcal{Z}$, where $i = 1, ..., d$. Since G operates on a vector of values similar to an AE, z_i is obtained from the 'reparametrization trick'. μ_i and σ_i values are utilized to sample $\boldsymbol{z}_i = \mathcal{N}(\mu_i, \sigma_i^2 \cdot \epsilon)$, where ϵ is a small value. The inclusion of ϵ is necessary to attribute the reconstruction error to the parameters of F. This ensures that the back-propagation can be applied to F, as the sampling of $\boldsymbol{z}$ is contained within a region defined by ϵ for a given $\boldsymbol{x}_n$.

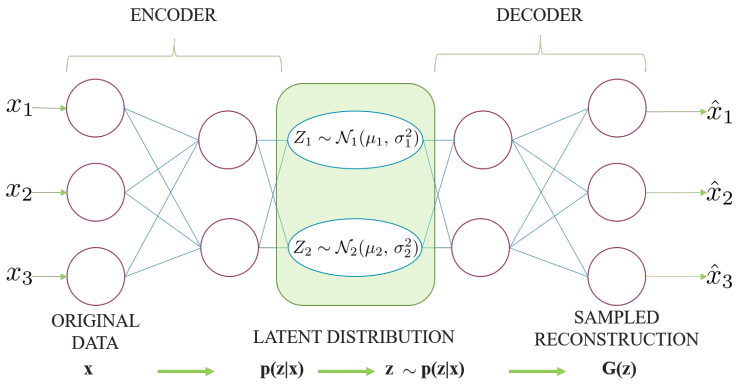


Figure I.22. Variational Auto-Encoder architecture using a FC-NN with a normal distribution at latent space.

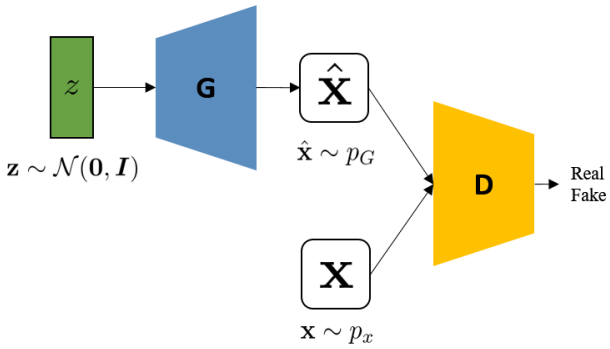


Figure I.23. GAN schema with inputs for G and D. The generator takes $\mathbf{z}$ vector to generate fake data set samples. The discriminator takes the fake and real samples to classify them as fake or real. G and D are DNNs with a combination of convolutional and dense layers.

Convolutional or dense layers can be indifferently used to code or decode data on AE or VAE. The type of data (images, time series, vector of labels) shall define the more suitable layers. Additionally, how the information of the data is intended to be structured on $\mathcal{Z}$ defines the types of layers. $\mathcal{L}$ in AE or VAE can also be adapted with additional terms to enforce desired characteristics on $\mathcal{Z}$.

The following sub-section talks about deep generative models. This method also applies representation learning concepts and introduces adversarial learning. A separate section is given to deep generative models since it is primarily used later in the methodology.

I.7.6 . Deep generative models

Many application of ML and DL has been evoked. This section develops the DL methods enumerated in Fig. I.19 as representation learning methods. However, a different application is focused on here: deep generative learning. Many architectures in the bibliography can be used as what we call data generators or simply *generators*.

Using DNNs as data generators started with the publication of [75]. The proposed Generative Adversarial Networks (GAN) method trains a neural network in the task of generating new unseen data for a data set $\mathbf{x} \in \mathcal{X}$, where $\mathbf{x}$ follows a data distribution p_x. A DNN architecture denoted by G is trained to take a noisy vector, denoted by $\mathbf{z} \in \mathcal{Z}$, as input to then generate an instance $\hat{\mathbf{x}} \in \mathcal{X}$, where $\hat{\mathbf{x}}$ follows a data distribution p_G. It is expected that $\hat{\mathbf{x}}$ 'looks like' a real instance on $\mathcal{X}$, but implementing a loss $\mathcal{L}$ to measure this is not a simple challenge.

The author proposed to use a *discriminator* to help construct $\mathcal{L}$. The discriminator D is also a DNN that is expected to discriminate between real samples from ($\mathbf{x} \in \mathcal{X}$) and fake samples $\hat{\mathbf{x}}$ from G. When the discriminator can not see any difference between real and fake samples, the generator succeeds in its generative task.

The loss for GAN is in construct with the generator and discriminator outputs in two parts: $\mathcal{L}_G$ and $\mathcal{L}_D$. D is a trainable functional (or DNN) that produces an output for each $\hat{\mathbf{x}}$ or $\mathbf{x}$, and it is expected for D to return 0 for the fake samples and 1 for the real samples. For instance, the binary output (scalar between 0 and 1) is used to compute the cross-entropy loss. D is the trained by $\mathcal{L}_D$ (Eq. I.18). In terms of statistics, it is called a two-sample test, and it is used to say if $\mathbf{x}$ and $\hat{\mathbf{x}}$ are drawn from the same distribution.

$$\mathcal{L}_D = \max_D \{\log D(\mathbf{x}) + \log(1 - D(\hat{\mathbf{x}}))\} \tag{I.18}$$

In contrast, G uses a noise input $\mathbf{z} \in \mathbb{R}^d$ as a source of randomness. $\mathbf{z}$ is drawn, for instance, from a normal distribution, so $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ where $\mathbf{0}$ is the zero matrix of dimension $(d \times d)$, and $\boldsymbol{I}$ is the identity matrix of dimension $(d \times d)$. G generates $\hat{\mathbf{x}}$ samples ($\hat{\mathbf{x}} = G(\mathbf{z})$) while following the objective

to fool D, so it classifies $\hat{\mathbf{x}}$ as real. For this aim, the loss $\mathcal{L}_G$ is used to train G by $\mathcal{L}_G$ (Eq. I.19). Both losses construct the global loss for the adversarial training of GANs.

$$\mathcal{L}_G = \min_G\{\log(1 - D(\hat{\mathbf{x}}))\} \tag{I.19}$$

The adversarial game between D and G (also called min-max game) is summarized by Eq. I.20, where $\mathbb{E}[\cdot]$ is the average when several samples of $\mathbf{z}$ and $\mathbf{x}$ are drawn from they distributions by batches.

$$\mathcal{L}_{GAN} = \min_G \max_D\{\mathbb{E}_{\mathbf{x}\sim p_x}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}[\log(1 - D(G(\mathbf{z})))]\} \tag{I.20}$$

$\mathcal{L}_{GAN}$ is then used by a gradient descent optimization algorithm (Subsection I.20) to update G and D parameters. The adversarial loss $\mathcal{L}_{GAN}$ formulation presented is not unique, but it can take several forms, like hinge loss [76] or Wasserstein distance [77], where the adversarial game is measured in different terms compared to the cross-entropy loss.

Additionally, the loss functions can also be adapted to different generative tasks. Generating realistic images, text, or voice is not the only ability for GANs. Some particular objectives can generate images with a given style [78, 79, 80]. Exploring $\mathcal{Z}$ space after G is trained is also an active research branch in generative networks. For instance, a possible objective is to use it as an interpolation space by finding correlations of $\mathbf{z}$ and so some real-world characteristics, like in [81] where the $\mathcal{Z}$ of realistic face generator DNN StyleGAN from [82] is explored to generate different pose or physical characteristics in a face, like age, gender, hair style, among others. Other examples of this approach can be found in [83, 84, 85]. GAN article is one of the pioneers of the generative model techniques, followed by the Deep Convolutional GANs (DCGAN) architecture by [86] that presents substantial improvements in data generation. Since its publication, several DNN schema have been developed as data generators. For instance, some approaches use the representation learning DNN architectures (AEs or VAEs in Subsection I.7.5.2) as generators. Even some architectures conceived for different proposes, such as the U-Net [87], have been adapted to be used as data generators. For this reason, it is possible to think of a new AI research field named generative models encompassing all these architectures.

Most recently, diffusion models [88] have been largely developed to enhance the generative capabilities of generative models [89]. This colloquially called 'new AI technology' is accessible now to the general public and shows amazing progress in realistic data generation. Large data sets such as CelebA, ImageNet, Audioset, or even text on the Internet have fed these architectures to be trained to generate high-quality images, text, or audio. Exploring this recent approach may be an interesting perspective for this work.

This thesis uses generative models throughout the methodology to generate NDT&E data. The architectures proposed are varied, ranging from AE, U-Net to a most complex Adversarial AutoEncoder (AAE) schema that borrows the GAN adversarial concept and DCGAN improvements to train an ensemble of DNNs. The objective in each of the following Chapters is not only to generate new NDT&E data but also to try to understand how DNN works on this task, and how they can be used to generate new significant data. The final objective is to use this to enhance automatic diagnostic, among other relevant techniques, so generating unlabeled data are not enough. For instance, the Basic GAN approach only cares about whether the generated samples look real. An example of this is the StyleGAN, which was conceived initially only to generate random realistic faces. In the NDT&E field, this approach needs to be pushed beyond to generate new labeled and informative data to be later deployed for enhanced diagnostics.

Before developing the specific generative architectures, some complementary concepts from ML and data generation are mentioned above since they are applied later in the Methodology.

I.7.7 . Transfer learning and domain adaptation on NDT&E

Transfer Learning Transfer Learning (TL) uses the concepts of source and target tasks and domains[90].

In ML, the algorithms learn from data to perform an expected task (e.g., classification, regression, segmentation). In some cases, it is possible to benefit from an algorithm already trained for a given task (a source task) to perform a different task, called a target task. Several TL techniques can be applied in this scenario to transfer the learned feature extractors (e.g., CNN) from a source task to a target task. For instance, the first layers of a trained NN for image classification can be used to train a regressor.

In other cases, collected data are separated into domains (source and target). The source and target domain present different characteristics, but share the same task. The target domain is often poorly accessible in terms of labels or samples.

Some NDT&E problems apply TL to take advantage of previously trained NN to facilitate the training of a new NN architecture [11, 12, 13] by reusing some layers.

I.7.8 . Domain adaptation and multi-fidelity data

Transfer learning can be effective when just a fidelity level is present during the training to achieve better performance, even when the data set is far from the in-situ data in terms of distribution and the data set is small compared to the complexity of the task. A succeeding approach from TL is the Domain Adaptation (DA) , suitable when more than one data fidelity is available. However, it commonly happens for NDT&E data to find that the better the fidelity is (closer to in-situ data), the less information about how these data were produced is available [14]. This is translated to a poorly labeled high-fidelity data set. To circumvent such an issue, DA mixed with generative or domain-adversarial strategies [16] have been tested to enhance DL algorithms for NDT&E inspections [17, 18, 19, 20]. DA applications can vary from large access of labeled data in each fidelity level to just one fidelity level with labels available [91].

In our scenario, the source domain can have its origins in a simulated NDT&E procedure, while the target domain is the experimental data where an inversion ML algorithm is intended to be deployed. In other words, two or more degrees of high-fidelity data are accessible during training (e.g., higher reliable simulation, experimental or in situ data).

I.8 . Main thesis contributions

The contributions of this thesis on the use of machine learning as an efficient tool for the generation and the exploitation of more realistic signals are presented in the following Chapters. The outcomes of this research have been structured accordingly into two differentiated parts. First, the challenges of multi-fidelity simulation by a surrogate model have been studied. Secondly, the developed models have been applied to perform ML-based diagnostics as applied to NDT&E.

More into detail, the first part of this analysis is dedicated to creating surrogate models that learn from data coming from NDT&E techniques. The simplest scenario is when only a fidelity level is available. The most accessible source of single-fidelity data is simulation: parametric simulations, FEM, among others. A particular challenge is presented when more fidelity levels are available. Any data source close to reality (e.g., from experience, in situ, or more reliable simulation) contains valuable information for a given study case on NDT&E. The question is how this information contained in multiple fidelity data sets can be exploited. The present work gives some insights and avenues to work in multi-fidelity data sets on NDT&E to later produce more data, enlarging the data in an informative way while it gets the closer possible to reality. Machine learning is the selected tool for surrogate model conception since it is a promising approach for massive and fast data generation. However, machine learning still presents some issues regarding generalization, explainability, and confidence that must be treated.

The second part objective is to use the newly generated data to enhance automatic diagnostics. The exploitation of the generated data for the inverse problems (e.g., diagnostic tools) may improve the performance on tasks such as regression (e.g., parameter estimation), classification (e.g., material microstructure inference), and statistical and performance analysis.

Fig. I.24 shows an overview of the thesis project and its principal parts. In the following chapters, both parts are developed together for specific NDT&E study cases. From Chapters II to IV, a NDT&E data set is introduced together with the ML methodology implemented. Some results to quantify the ac-

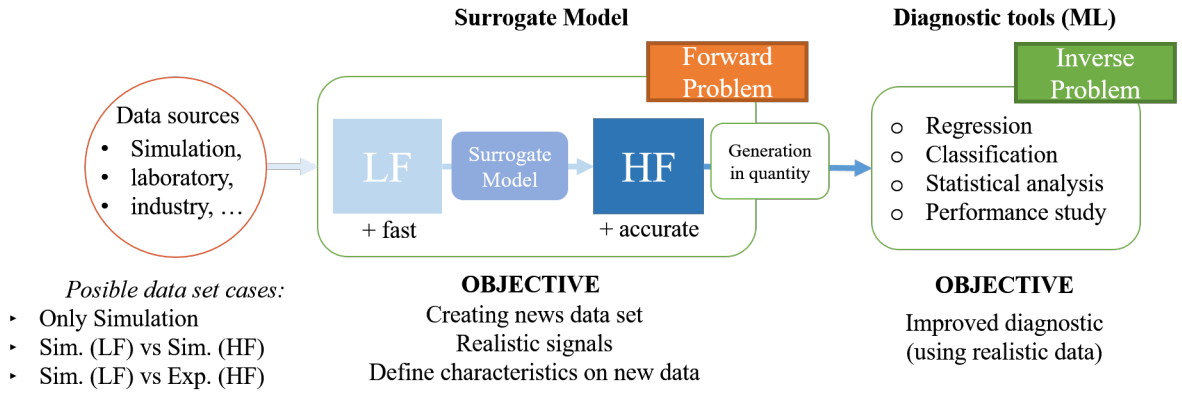


Figure I.24. Overview of the thesis project. LF: low fidelity. HF: high-fidelity. ML: Machine-learning algorithms. The forward problem focuses in the development of $\mathcal{M}$, while the inverse problem shows an application of $\mathcal{M}$ in diagnostic.

curacy of the surrogate model are exposed, and when it is pertinent, an application of the newly generated data are assessed for an inverse problem.

- In Chapter II, we implement the surrogate model via a conditional AutoEncoder architecture (forward problem) for an ECT simulated data set, where a set of parameters $\mathbf{p}$ is used to condition the DL architecture. The aim is to build a surrogate model $\mathcal{M}$ to replace $\mathcal{F}$. Later, $\mathcal{M}$ is deployed in statistical analysis (inverse problem) for the ECT inspection. The results were published in [92].

- In Chapter III, we apply a conditional U-Net architecture as a surrogate model for a multi-fidelity UT TFM imaging data set. The data set contains simulated and experimental acquisitions. The direct problem used a similar conditioning to the previous Chapter, by the parameters $\mathbf{p}$ from the simulated data. $\mathcal{M}$ is a deterministic mapping from simulation fidelity data to experimental fidelity data. The results were published in [93].

- In Chapter IV, we use the approach in Chapter IV to develop a stochastic generator from a multi-fidelity UT TFM imaging data set. $\mathcal{M}$ generates multiple experimental samples from a simulation sample, by learning from the experimental examples in the data set. The inverse problem aimed here is the inversion of $\mathbf{p}$ by a DL, an additional neural network with the generated data from $\mathcal{M}$. The results were published in [94].

Some of the implemented ML learning techniques and architectures and some NDT&E techniques are not described in this introduction, but they are described in the following Chapters to avoid an extensive and too abstract introduction.

II - Development of a deep learning framework in an eddy current testing simulated data set as surrogate model for efficient statistical studies

About this section

In this chapter, we present a *supervised* deep learning framework in a *simulated data set* to be used as a surrogate model for efficient statistical studies. NDT&E simulation can be expensive in terms of time calculation and some diagnostic studies require large amount of data to be assessed. This first approach is intended to find a deep learning method suitable for the development of a surrogate model on NDT&E in paricular contexts. The development framework is exploited later for a multi-fidelity case.

II.1 . Introduction

In Nondestructive Testing an Evaluation (NDT&E) field, the vast improvement in numerical simulation tools in terms of efficiency is mainly due to the increases of computational power on standard PCs and through the use of distributed and cloud computing resources. Nevertheless, this progress has just partially mitigated the computational efficiency issues that one faces in performing very demanding statistical studies such as Global Sensitivity Analysis (GSA) [95, 96], Model-Assisted Probability of Detection (MAPOD) [97, 98, 99], stochastic optimization [100], inversion[101, 102], etc. In order to decrease the computational burden without degrading the quality of the results, metamodels (also known as surrogate models) are employed to replace the "true" physics-based model for a given set of parameters (or factors). Loosely speaking, a metamodel can be defined as a mathematical function mapping the parameter space versus the measures space, where measurements can be scalar or vector-valued quantities. In the context of machine learning and statistical methods, the most employed metamodels rely on shallow learning methods (i.e., kernel machines) [101, 103, 104], statistical methods (i.e., Gaussian process, polynomial chaos expansion, etc.) [105, 106], ensemble methods (i.e., random forest, extreme gradient boosting, etc.) or Deep Neural Network (DNN) architectures [107]. All these methods exploit a supervised learning framework where the inputs correspond to a labeled target (or, equivalently, the measurement output).

That is, the use of a pre-computed data set (or database), which contains a collection of input parameters and output signals, is generated during an off-line phase based on physics-based simulations, and a specific ML model is fit on the data available. In the second phase, referred to as the online phase, a metamodel is used to generate output signals on a set of unseen input parameters. The metamodel acts as a black-box quasi-real-time replacement of the complete forward solver. In this way, it can be plugged in a transparent way within any kind of algorithm involving the use of a physical model for speeding-up considerably the computational efficiency.

In the NDT&E community, the use of metamodels applied to GSA [96, 108] and MAPOD [97] has been extensively investigated in the past based on different configurations. Nevertheless, using metamodels based on DNN remains marginal compared to kernel machines or statistical methods. One of the main reasons for considering DNN architectures over the aforementioned approaches is the higher scaling efficiency in large and complex data sets. Therefore, DNN-based methods are among the most suitable regressor to be applied when the cardinality of the inputs is very large.

This chapter focuses on an Eddy Current Testing (ECT) inspection scenario driven by twelve parameters used to describe probe, specimen and defects positional or geometric characteristics. Simulations have been performed into the CIVA-DS application (i.e., the database generation and ML-focused package of the CIVA commercial platform) [109], efficiently addressing the generation of data sets for a wide

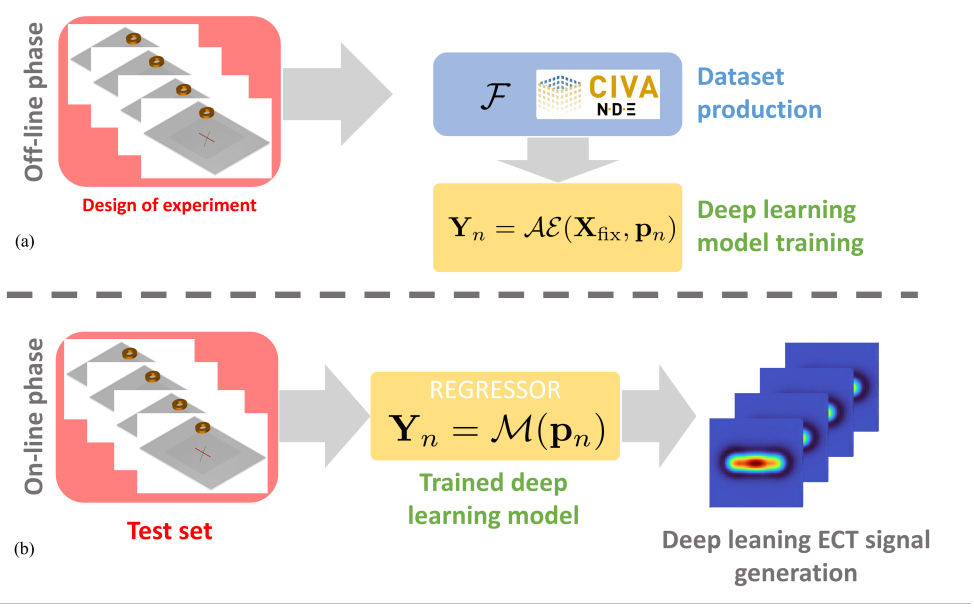


Figure II.1. Sketch of the supervised deep neural network schema employed divided as (a) offline phase where a $\mathcal{AE}$ deep learning model is trained and (b) online phase where the deep learning model is used as regressor $\mathcal{M}$. The variables $\mathbf{Y}_n, \mathbf{X}_{fix}, \mathbf{p}_n$ of this schema are difined in the next sections.

range of NDT&E problems regardless of the method considered. A DNN is trained over this data set to later be used to efficiently compute the GSA and the Feature Importance (FI).

This chapter is structured as follows. In Section II.2, we present the supervised framework adopted in our studies with a particular emphasis on the DNN architecture developed. In Section II.3, we introduce the GSA methods employed in this work along with the Shapely additive explanation (SHAP) [110] used to analyze the impact/importance of the parameters on the model outputs, named the FI method in this work. Subsequently, the DNN performance is assessed in Section II.4, and the GSA and FI results are discussed in Section II.4.2. The chapter ends with the conclusions and future perspectives.

II.2 . Supervised DNN regression schema applied to ECT signals

In the last decade, fast regression models (i.e., a metamodel or surrogate model) based on pre-computed databases made of homogeneous collections of NDT&E inspection signals (e.g., A-scan, B-scan, C-scan etc. signals) and/or engineered extracted features from inspection signals (e.g., peak values, time of flight, etc.) and the associated inspection parameters have been increasingly employed to speed-up the computational time of practical studies for assessing the performance of the inspection procedure such as the MAPOD framework [96, 99, 111, 112, 106, 113]. More recently, the use of surrogate models has been applied to enable the almost real-time application of global sensitivity analysis studied based on statistical distributions. Nevertheless, the use of metamodels based on DNN architectures is not deeply studied in the context of NDT&E based on ECT signals if applied to the efficient calculation of GSA indexes or FI ranking based on SHAP values.

II.2.1 . Simulated ECT data-set

We define a database $\mathbb{D}$ containing a set of N couples (or samples) as $\mathbb{D} = [(\mathbf{p}_1, \mathbf{Y}_1), (\mathbf{p}_2, \mathbf{Y}_2), ..., (\mathbf{p}_N, \mathbf{Y}_N)]$, where the vector associated with the i-th sample writes as $\mathbf{p}_n = [p_1, p_2, ..., p_D]$ with D input parameters size such that $\mathbf{p}_n \in \mathbb{R}^D$ and the corresponding target vector made by M element is,

$$\mathbf{Y}_n = [\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_N].$$

$$\mathcal{AE}(\mathbf{p}_n, \mathbf{X}_{\text{fix}}) = (\mathcal{S}_3 \circ \mathcal{S}_2 \circ \mathcal{S}_1 \circ \mathcal{D}_2 \circ \mathcal{D}_1 \circ \mathcal{LS} \circ \mathcal{E}_3 \circ \mathcal{E}_2 \circ \mathcal{E}_1)(\mathbf{p}_n, \mathbf{X}_{\text{fix}}) \tag{II.1}$$

$$\begin{aligned}
\mathcal{E}_i(\mathbf{p}_n, \mathbf{Y}_{\mathbf{f_{i-1}}n}) &= \mathcal{L}_{pool} \circ \sigma(\mathcal{L}_{\text{p}}(\mathbf{p}_n)) \circ \sigma(\mathcal{L}_{\text{p}}(\mathbf{p}_n, \mathbf{Y}_{\mathbf{f_{i-1}}n})) + \mathcal{L}_{pool} \circ \mathcal{L}_{conv}(\mathbf{Y}_{\mathbf{f_{i-1}}n}) \\
\mathcal{L}_{\text{p}}(\mathbf{p}_n, \mathbf{Y}_{\mathbf{f_{i-1}}n}) &= \mathcal{L}_{pST}(\mathbf{p}_n) \circ \mathcal{L}_{FiLM}(\mathbf{p}_n) \circ \mathcal{L}_{conv}(\mathbf{Y}_{\mathbf{f_{i-1}}n}) \\
\mathcal{LS}(\mathbf{Y}_{\mathbf{f_{i-1}}n}) &= \sigma(\mathcal{L}_{\text{p}}(\mathbf{p}_n)) \circ \sigma(\mathcal{L}_{\text{p}}(\mathbf{p}_n, \mathbf{Y}_{\mathbf{f_{i-1}}n})) + \mathcal{L}_{conv}(\mathbf{Y}_{\mathbf{f_{i-1}}n}) \\
\mathcal{D}_i(\mathbf{Y}_{\mathbf{f_{i-1}}n}) &= \mathcal{L}_{up} \circ \sigma(\mathcal{L}_{IN} \circ \mathcal{L}_{conv})(\mathbf{Y}_{\mathbf{f_{i-1}}n}) + \mathcal{L}_{up} \circ \mathcal{L}_{conv}(\mathbf{Y}_{\mathbf{f_{i-1}}n}) \\
\mathcal{S}_i(\mathbf{Y}_{\mathbf{f_{i-1}}n}) &= \sigma_{TanH}(\mathcal{L}_{IN} \circ \mathcal{L}_{conv})(\mathbf{Y}_{\mathbf{f_i}n})
\end{aligned} \tag{II.2}$$

These target or output values are obtained by applying a deterministic forward operator $\mathcal{F}$ on the set of input parameters $\mathbf{p}_n$, i.e., $\mathbf{Y}_n = \mathcal{F}(\mathbf{p}_n)$. More generally, we can define $\mathcal{F} : \mathbb{R}^{1\times \text{D}} \rightarrow \mathbb{C}^{\text{M}_\text{x}\times\text{M}_\text{y}}$ and therefore $\mathbf{Y}_n \in \mathbb{C}^{\text{M}_\text{x}\times\text{M}_\text{y}}$ where $\mathcal{F}(\mathbf{p}_n)$ is obtained via a CIVA solver simulation based on the integral approach [34, 114], governed by the Equations I.1, I.2, I.3, I.4, I.5 from Chapter I. Different database-building strategies have been developed in the literature; some of them rely on fixed sampling schema of the parameter space, while others aim to increase the parsimony in terms of the number of calls to the forward solver without degradation of the accuracy in metamodel results. In this work, we employ a database sampling-based one-shot strategy where the parameter space has been sampled based on Latin hyper-cube sampling.

The metamodel ($\mathcal{M}$) employed in this work is based on a DNN architecture [107] tailored for complex-valued ECT signals based on a C-scan inspection procedure. Based on the tight analogy of ECT signals with images (indeed, up to some extent, ECT measurements can be seen as hyper-spectral images with two channels), we developed a specific encoder-decoder (AutoEncoder-like (AE)) architecture based on $2D$-convolutional layers alternated to PReLU activation function [115] and average pooling layers. Furthermore, to enable the regression capability based on the variation of input parameters only (i.e., see (Eq. II.1)), we accounted for the Feature-wise Linear Modulation (FiLM) [116] and a modified Spatial Transformer (ST) layers [117], name here as parametric ST (pST).

II.2.2 . Conditional auto-encoder-like architecture

The AE-like architectures in the bibliography may vary regarding the objective of the DNN and the type of data used to train it. For AE-type neural networks, the layers before the middle latent space (LS) are tailored to the expected task during the on-line phase (Fig. II.1 a)); e.g., variational AEs rely on fully connected layers to infer distribution parameters, long short-term memory encoders are used to embed the useful information to infer future states of an input. In our case, the feature extraction procedure performed by the architecture is based on $\mathbf{p}_i$ at every layer of the encoder. The encoder use a fixed 2-channel noisy input $\mathbf{X}_{\text{fix}}$ with the same dimension of $\mathbf{Y}_i$. This may lead to an LS structured by the parameters used during the data production. The expected behavior during the on-line phase is a coherent ECT generation piloted by $\mathbf{p}_i$. It is worth to be noticed that a structured LS based on the knowledge of input parameters enhance a better insight into the inner working mechanisms of the NN architecture as well as assures that the underlying physics is injected and preserved into the architecture in the different layers. A sketch of the architecture developed in this work is represented in Fig. II.2. The encoder, bottleneck and decoder blocks are highlighted in yellow ($\mathcal{E}_i$), green ($\mathcal{LS}$) and blue ($\mathcal{D}_i$), respectively. Additional synthesis layers ($\mathcal{S}_i$), displayed in purple, are added to improve reconstruction accuracy. The encoder and decoders count with Res-Net-like skip-connections to help the convergence for the training. Spatial dropout layers are added at every block to promote the independence of per-channel features.

We denote with the subscript n a sample that is forwarded in $\mathcal{AE}$ (Eq. II.1). $\mathbf{Y_f}$ is the features extracted by the layers in $\mathcal{AE}$. i is the number of the layers in $\mathcal{AE}$ whose input is the $\mathbf{Y_f}$ from the precedent layer $i-1$, together with the $\mathbf{p}_n$ corresponding to the sample $\mathbf{Y}_{\mathrm{n}}$. The inner layers operators at each layer (Eq. II.2) are identified as $\mathcal{L}_{\mathrm{p}}$the parametric layer triplet (composed by the parametric spatial transformer$\mathcal{L}_{pST}$, the feature-wise linear modulator $\mathcal{L}_{FiLM}$ and a 2D convolution $\mathcal{L}_{conv}$. σ denotes the PReLU activation, while σ_{TanH} is a tangent hyperbolic activation. $\mathcal{L}_{up}$and $\mathcal{L}_{pool}$ represent the up-sampling and average pooling operators. The encoder normalizes its convolution output by an instance normalization $\mathcal{L}_{IN}$ [118].

A Mean Square Error (MSE) loss is used to train the architecture. The neural networks optimized the reconstruction of $\mathbf{Y}_n$ (2-channel input) while learning how the FiLM and pST transformations are conditioned by $\mathbf{p}_n$. Those two layers are schematically represented in II.2(b) and II.2c, where $\mathbf{Y}_{\mathbf{f_{preced}}\,n}$ are the extracted feature by any precedent layer ($\mathcal{L}$).

For any n-th sample $\mathbf{p}_n$, the pST layer, placed after the i-th Res-Net block, uses a dense layer to infer $\boldsymbol{\phi} : \mathbb{R}^{6\times C_k}$ from $\mathbf{p}_n$ to built $T_{\boldsymbol{\phi}} : \mathbb{R}^{3\times 3\times C_k}$, where C_k are the channel number of the input feature map $\mathbf{Y}_{\mathbf{f_{preced}}\,n} \in \mathbb{R}^{\mathrm{W_i}\times \mathrm{H_i}\times \mathrm{C}_k}$. $\boldsymbol{\phi}$ and $T_{\boldsymbol{\phi}}$ for each C_k are built as

$$\begin{gathered}\boldsymbol{\phi}_{C_k}(\mathbf{p}_n) = [\phi_{1_{C_k}}(\mathbf{p}_n),\ \phi_{2_{C_k}}(\mathbf{p}_n),\ \phi_{3_{C_k}}(\mathbf{p}_n),\\ \phi_{4_{C_k}}(\mathbf{p}_n),\ \phi_{5_{C_k}}(\mathbf{p}_n),\ \phi_{6_{C_k}}(\mathbf{p}_n)],\\ T_{\boldsymbol{\phi}_{C_k}}(\boldsymbol{\phi}_{C_k}) = \begin{bmatrix} \phi_{1_{C_k}} & \phi_{2_{C_k}} & \phi_{3_{C_k}} \\ \phi_{4_{C_k}} & \phi_{5_{C_k}} & \phi_{6_{C_k}} \\ 0 & 0 & 1 \end{bmatrix},\end{gathered} \tag{II.3}$$

to be applied as an affine transformation per channel C_i to the feature map coordinates of $\mathbf{Y}_{\mathbf{f_{preced}}\,n}$as follows,

$$\mathrm{coord}(\mathbf{Y}_{\mathbf{f-st}_n}) = T_{\boldsymbol{\phi}}(\boldsymbol{\phi}(\mathbf{p}_n)) \odot \mathrm{coord}(\mathbf{Y}_{\mathbf{f_{preced}}\,n})\ . \tag{II.4}$$

The resulting coordinates from Eq.II.4 are used to compute a transformed feature map $\mathbf{Y}_{\mathbf{f_{preced}}\,n}$. To do that, the pixel intensities in $\mathbf{Y}_{\mathbf{f_{preced}}\,n}$ are used together with the source coordinate grid $\mathrm{coord}(\mathbf{Y}_{\mathbf{f_{preced}}\,n})$ to get the pixel values for the new target grid $\mathrm{coord}(\mathbf{Y}_{\mathbf{f-st}_n})$. A bi-linear interpolation is applied with the four close neighbors from the source grid. As a result, $\mathbf{Y}_{\mathbf{f-st}_n}$ is a representation of $\mathbf{Y}_{\mathbf{f_{preced}}\,n}$ after the learned an affine transformation is applied at each channel. The operation denoted by $\odot$ performs a matrix multiplication each set of (x, y) coordinates at $\mathbf{Y}_{\mathbf{f_{preced}}\,n}$ for each channel. The whole pST operation (Eq. II.3 and II.4) is represented by $\mathcal{T}$from now-on, witch is applied to$\mathbf{Y}_{\mathbf{f}_{\mathrm{i}}\,n}$ as follows,

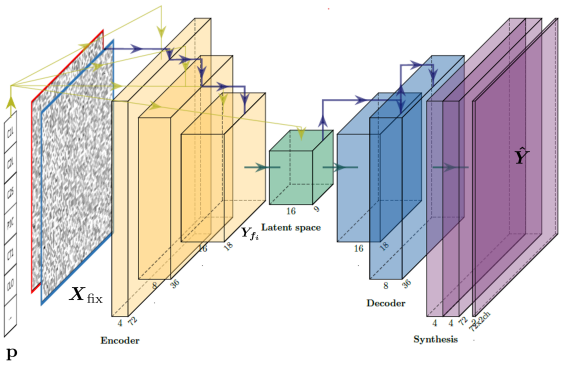
$$\mathcal{L}_{\mathrm{pST}}(\mathbf{p}_n, \mathbf{Y}_{\mathbf{f_{preced}}\,n}) = \mathcal{T}(\mathbf{p}_n) \circ \mathbf{Y}_{\mathbf{f_{preced}}\,n} \tag{II.5}$$

Analogously to pST, the FiLM uses a dense layer to infer $\beta(\mathbf{p}_n) : \mathbb{R}^{C_k}$ and $\gamma(\mathbf{p}_n) : \mathbb{R}^{C_k}$ from $\mathbf{p}_n$ to apply, then

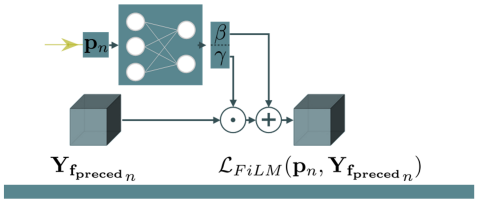
$$\mathcal{L}_{FiLM}(\mathbf{p}_n, \mathbf{Y}_{\mathbf{f_{preced}}\,n}) = \gamma(\mathbf{p}_n) \cdot \frac{\mathbf{Y}_{\mathbf{f_{preced\,n}}} - \mathbb{E}[\mathbf{Y}_{\mathbf{f_{preced\,n}}}]}{\sqrt{\sigma^2[\mathbf{Y}_{\mathbf{f_{preced\,n}}}] + \epsilon}} + \beta(\mathbf{p}_n), \tag{II.6}$$

where $\mathbb{E}\,[\cdot]$ and $\sigma^2\,[\cdot]$ represent the empirical average and variance, respectively. ϵ is a small value used for numerical stability, normally set to 1×10^{-3}.

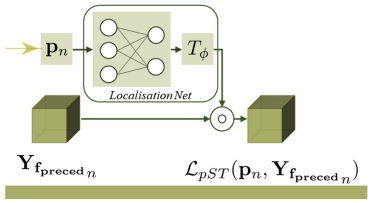
In order to use the AE-like architecture as a parametric regressor, we fix the input to $\mathbf{X}_{\mathrm{fix}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{I}, \mathbf{0}\ \in \mathbb{R}^{M_x\times M_y\times 2}$ during the training and the test phases. In contrast, using $\mathbf{Y}_n$ as input during the training like in a pure AE architecture, may not result in a surrogate model $\mathcal{M}(\mathbf{p}_n)$, since an input is needed during the inference. In the Appendix VII we comment how the pST and FiLM layers acts when a traditional AE is trained. The first observation is that the choice of the input during the inference introduce a bias in the generation (see Appendix Fig. VII.1 for more details). Secondly, it is the fact that the encoder do not use the particular image features of the input to create the latent space (see Appendix



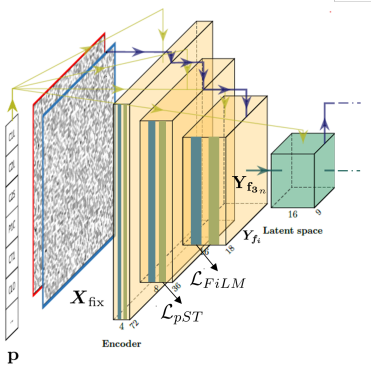
(a) $\mathcal{AE}$ blocks disposition



$(b)\mathcal{L}_{FiLM}$ operator



$(c)\mathcal{L}_{pST}$ operator



(d) Detail for the FiLM and pST conditional layers in the encoder

Figure II.2. In (a), the convolution AutoEncoder-like (AE) architecture was modified to have a parametric conditional input (yellow arrows). The architecture reconstructs the ECT image, guided by the parameters. Yellow blocks contain parametric (b) Spatial Transformers layers (pST) and (c) Fidelity Layer Modulator (FiLM) for conditioning. The green block in (a) is the latent representation of the input image given by the encoder. The encoder and synthesis layers generate a conditional image. The blue arrows represent residual connections. In (d), a detail for how the conditional layers are set in the encoder. Figure created with [119].

Fig. VII.2 for more details). Instead, the encoder takes any image at the input together with $\mathbf{p}_n$ to create the necessary encoding to get $\mathbf{Y}_n$, even when the input features is far from the desired $\mathbf{Y}_n$. This observation takes us to the conclusion that fixing the input to a arbitrary input such as, as instance, a noisy spatially correlated input may improve the generation during the test. The interest of this relies on avoiding the bias traditional AE introduced by the choice of the input. This convolution feed-forward encoder-decoder is allowed to create a small parameter metamodel that, at the same time, allows us to reduced to a minimum the number of training samples, competing with, as example, the number of parameters for a multi dense layer architecture to map $\mathbf{p}_n$ to $\mathbf{Y}_n$.

Behind the choice of this architecture is the underlying assumption that a set of spatial transformations (Eq. II.5) exists to generate the ECT images from a fixed input, and these transformations depend on the simulation parameters. Similarly, the distribution of each ECT image depends on the input parameters; this is learned by Eq. II.6. Both transformations are a function of the simulation parameters. For the first assumption, the coil tilt is an example of a spatial transformation required to produce different coherent outputs from a fixed image input but a changing parameter input $\mathbf{p}_n$. For instance, the main feature in Fig. II.5(b)-left, in contrast to Fig. II.5(b)-right, presents a spatial scale learned by the pST, among others. Similarly, the two images present different distributions learned by the FiLM.

II.2.3 . Metamodel validation metrics

To quantitatively evaluate the metamodel prediction accuracy of multivariate ECT signals, we shall employ three different metrics. The first one is the normalized root mean squared error (NRMSE), defined as

$$NRMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\frac{\left(\mathcal{F}(\mathbf{p}_n) - \mathcal{M}(\mathbf{p}_n)\right)^2}{\mathbf{Y}_{max} - \mathbf{Y}_{min}}}, \tag{II.7}$$

where $\mathcal{F}(\mathbf{p}_n) = \mathbf{Y}_n$ and it stands for the n-th true value to be estimated and $\mathcal{M}(\mathbf{p}_n)$ is the associated estimation. $\mathbf{Y}_{max}$ and $\mathbf{Y}_{min}$ correspond to the maximum and minimum ECT signals values among Nvalues considered, respectively. To estimate the fit of the predicted ECT signals against the ground truth (GT), we employ the correlation coefficient (R^2) defined as

$$R^2 = 1 - \sum_{n=1}^{N}\frac{\left(\mathcal{F}(\mathbf{p}_n) - \mathcal{M}(\mathbf{p}_n)\right)^2}{\left(\mathcal{F}(\mathbf{p}_n) - \bar{\mathbf{Y}}\right)^2}, \tag{II.8}$$

where $\bar{\mathbf{Y}}$ is the mean value of the ECT signals among the N samples considered. Lastly, the mean normalized Frobenius norm ($\|\cdot\|_F^2$) error (MNFE) has been calculated as

$$MNFE = \frac{1}{N}\sum_{n=1}^{N}\frac{\|\mathcal{F}(\mathbf{p}_n) - \mathcal{M}(\mathbf{p}_n)\|_F^2}{\|\mathcal{F}\{\mathbf{p}_n\}\|_F^2}. \tag{II.9}$$

II.3 . Metamodel-based sensitivity analysis and feature importance studies applied to ECT signals

In real-case scenarios, the agreement between experimental data and simulated results depends on the capability to master the whole acquisition chain (i.e., probe position, dimensions, knowledge of specimen characteristics, measurement noise, etc.). That is, in operative conditions, the measurement signals are impacted by 'hidden' factors that cannot be directly measured. In order to have a better insight into the impacts of these factors on measurement, one can consider them along with the driving factors that are supposed to be 'known', i.e., flaw size, position, etc., in a sensitivity analysis or FI ranking framework.

The main goal of global sensitivity analysis [95] consists on identifying and ranking the set of parameters that impact the model output variability across the entire data set variability. In a nutshell, the underlying concept is that the bigger the sensitivity of a parameter is, the higher its influence on the output is. In this section, we introduce the GSA method variance decomposition based on Sobol' indices

[120] and the moment-independent GSA method based on δ-sensitivity measure (or indexes), which is commonly employed for ranking and screening purposes [95]. In the ML research community, the ranking of the most important feature is performed through FI methods. In this section, we focused on the use of SHAP values[110] as a FI method to be applied to the analysis of the most important factors driving the ECT signals measurements.

Sobol' indices provide a quantitative measure of the contribution of individual input variables or combinations of variables to the overall variability in the model output. There are two main types of Sobol indices: first-order indices and total-order indices.

First-order indices represent the contribution of each individual input variable to the output variability when considered in isolation, without accounting for interactions with other variables. Total-order indices account for both the direct effect of a variable and its interactions with other variables. They measure the total contribution of a variable to the output variability, including both independent and interactive effects.

Regardless of the use of GSA or FI methods for ranking purposes, the computational burden associated with their calculation makes the use of metamodels mandatory for performing the sensitivity analysis in an acceptable amount of time. Referring to GSA, one can show that the first-order Sobol' and the total order indices of the i-th factor are given by [120, 121] $S_i = \frac{V_i}{\mathbb{V}ar(\mathcal{M}(\mathbf{p}))}$ and $S_{T_i} = 1 - \frac{\mathbb{V}ar_{\mathbf{p}_{\sim i}}(\mathbb{E}(\mathcal{M}(\mathbf{p})|p_{\sim i}))}{\mathbb{V}ar(\mathcal{M}(\mathbf{p}))}$, receptively. S_i and S_{T_i} represent the main effects of the i-th variable alone and the effects of the i-th variable and its interactions with the other variables, respectively. Where $\mathbb{V}ar(\cdot)$ represents the variance, $\mathbb{E}(\cdot)$ the expectation with $V_i = \mathbb{V}ar_{p_i}\left(\mathbb{E}_{\mathbf{p}_{\sim i}}\left(\mathcal{M}(\mathbf{p})|p_i\right)\right)$ with the subscript '$\sim$' identifies the left out index factor in the calculations [121].

It is worth to be mentioned that Sobol' indices capture the overall behavior of uncertainties when the variance of the outputs represents sufficiently (i.e., it is a good proxy for its estimation). δ-sensitivity measure is used to compute GSA indices based on the variation of conditional and unconditional probability density functions. This sensitivity analysis method is particularly suitable in the presence of a correlation between parameter and when the distribution of the outputs are highly skewed or multimodal. δ-importance (or δ-sensitivity) measure for the i-th factor is defined as [122] $\delta_i = \frac{1}{2}\mathbb{E}_{p_i}\left[s\left(p_i\right)\right]$ with

$$\mathbb{E}_{p_i}\left[s\left(p_i\right)\right] = \int f_{p_i}\left[\int \left|f_Y\left(\mathcal{M}(\mathbf{p})\right) - f_{Y|p_i}\left(\mathcal{M}(\mathbf{p})|p_i\right)\right| dy dp_i\right],$$

called inner statistic or inner separation, represents the area enclosed between the conditional ($f_{Y|p_i}$) and unconditional (f_Y) model output densities obtained for a particular value of p_i. This means that in the case of $f_{Y|p_i}$ is equal to f_Y, removing the uncertainty on p_i does not affect the distribution of the output; thus, the i-th input does not impact the output of $\mathcal{M}(\mathbf{p})$.

Feature importance methods can be used in order to rank the most impacting inputs onto the model outputs. Among the wide set of FI methods developed by the ML scientific community, in this work, we adopt SHAP [110], which is a method to explain predictions based on the use of Shapley values in the case of coalition game theory. Furthermore, SHAP values enable access to both global estimations of FI as well as to a qualitative assessment of the impact of inputs onto the model output. Loosely speaking, Shapley values express how the predictions are homogeneously distributed among the features (i.e., the parameters) and in the framework of SHAP, Shapley values are calculated as an additive feature attribution method as $g\left(z'\right) = \phi_0 + \sum_{j=1}^{M}\phi_j z'_j$with $g\left(\cdot\right)$ being the explanation mode, z' z' is the so-called coalition vector (i.e., the features/inputs considered), ϕ_j the Shapley value for the j-th feature and $\phi_0 = \mathbb{E}\left[f\left(z\right)\right]$ with $f(\cdot) = \mathcal{M}(\mathbf{p})|z$.

II.4 . Results

In this section, we analyze the results obtained for the proposed DNN-based metamodel schema once applied to ECT signals based on an inspection problem parameterized by coil, specimen and crack(s)

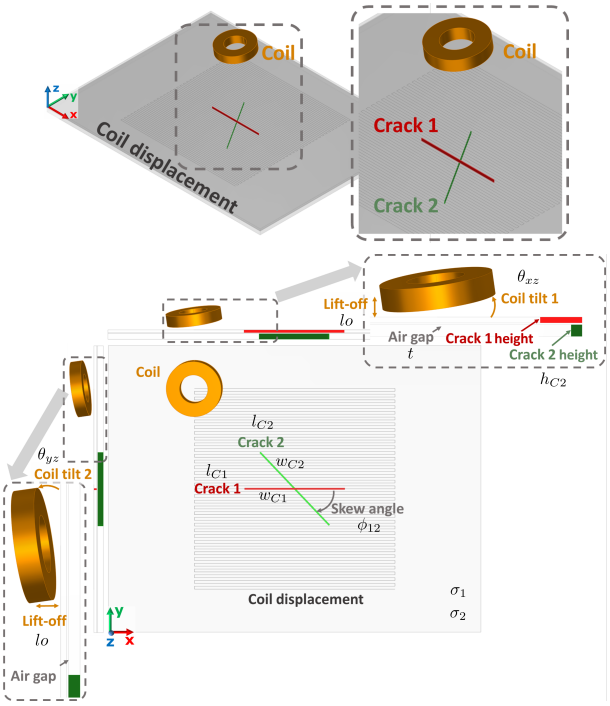


Figure II.3. ECT inspection configuration. Two isolated plates with a linear crack in each plate. A coil follows a scanning path to produce imaginary and real pixel maps. The $3D$ view and the associated three main orthogonal projections are shown.

parameters. The synthesis of ECT signals has been applied to the fast calculation of global sensitivity analysis (GSA) indices and feature importance (FI) calculations.

II.4.1 . Generative DNN-based metamodel performance

Referring to the inspection case shown in Fig. II.3, a suitable database has been built accounting for twelve parameters involving the cracks, probe and specimen parametrization. The forward solver simulations, based on the integral equation method [34, 114], have been performed by CIVA software [109]. More into detail, crack 1 length (l_{C1}) was made varying $[25.0; 35.0]$ mm, and its width (w_{C1}) takes values in the interval $[0.05; 0.5]$ mm. Crack 2 length (l_{C2}), width (w_{C2}) and height (h_{C2}) were varied in the ranges of $[25.0; 35.0]$ mm, $[0.05; 0.5]$ mm and $[0.25; 2.0]$ mm, respectively. The skew angle between the two cracks (ϕ_{12}) takes values between $[0.0; 110.0]$ deg. The coil lift-off (lo), tilt in the xz-plane (θ_{xz}), tilt in the yz-plane (θ_{yz}) assumed values in the range of $[0.05; 0.8]$ mm, $[0.0; 8.0]$ deg and $[0.0; 8.0]$ deg, respectively. Concerning the specimen, both plates conductivity (σ_1 and σ_2) varied between $[16.0; 22.0]$ MS/m along with the air gap thickness between the two plates (t) ranging as $[0.03; 0.3]$ mm.

The acquisition are done at the frequency of 1.5 kHz and the plates have 17 MSiemens for electrical conductivity.

For simulating the inspection procedure, a C-scan (i.e., a 2D map) made by $M_x \times M_y = 72 \times 72$ points centered on the cracks zone has been considered for simulations. Therefore, each sample within the database contains 5184 complex-valued measurement points corresponding to the coil impedance variation signal (ΔZ). The database has been sampled based on a Latin hyper-cube sampling schema made of 5 k items and then split into training, validation and test sets by choosing 1.25k, 625 and 2.5k samples, respectively. The generative DNN architecture introduced in Section II.2 has been trained with a learning rate equal to $1e^{-3}$ based on ADAM optimizer [123] with a batch size equal to 128. The training procedure ended after 5 k epochs in about 4 h on a GPU cluster equipped with one NVIDIA HGX A100 graphic card.

The DNN metamodel performance has been assessed on the whole set of 2.5k test samples accord-

ingly to the metrics provided in equations II.7, II.8 and II.9, obtaining error values as 0.007, 0.92 and $3.7e-9$, respectively. Qualitative comparisons of results based on the test set samples are given for the real and imaginary parts of the signal in Fig. II.4 and Fig. II.5. Fig. II.6 shows an example of generation by different input parameter changes. A good agreement has been observed between DNN prediction and ground truth (GT) all over the whole set of test. Therefore, one can conclude that the errors introduced by metamodel predictions of ECT signal are negligible for its exploitation for ranking purposes by GSA indices and FI calculation. Furthermore, from the point of view of computational performance, the test set predictions based on the whole test set take about $1.25\,\mathrm{s}$ on a PC equipped with an Intel Xeon CPU @3.70 GHz and a QUADRO RTX GPU 6000, which is a non-negligible speed-up in computational time efficiency if it is compared to the about $40\,\mathrm{seg}$ needed to compute one forward solver calculation.

II.4.2 . A deeper insight into the generative DNN metamodel procedure

Hereafter, we provide some insight into how the DNN learns to generate data. We explored the latent space block output to have access to the analysis of the operation of the encoder. Toward this end, the t-SNE manifold projection [70] was used to obtain a compact 2D visual representation of the latent space across the entire data set (i.e., the whole 5k samples).

Fig.II.7 represents a scatter plot of the LS space. Every point is located on the 2D manifold by its coordinates (LS_1, LS_2) based on the t-SNE with an initialization method relying on the principal component analysis (PCA) [124]. As a result, each point can be connected to a parameter value and properly visualized based on the 'color bar dimension'. In the analysis of the manifold, one can clearly appreciate that hierarchical arrangements appear as shown in Fig.II.7 $a)-b)$, whereas a less pronounced order is observed for the lift-off parameter. The observations demonstrated that inner features are structured by parameters and that the generative model developed focused first (i.e., encoded or extracted features) on the spatial information (i.e., the ECT flaw signature on the C-scan) associated with ϕ_{12} and l_{C1} and then to lo. Such behavior is possible since the input vector can produce both previously unseen samples as well as other samples found in the data collection. Furthermore, the projection demonstrates how the encoder features are altered in a coherent direction during the generation by regression.

A well-structured latent space does not imply a good reconstruction, but if the hierarchy found in the projection is coherent with the training data set structure, it may indicate that the DNN has captured correctly the spatial features of the input images and so the empty space can be filled in a coherent manner. In other words, the DNN who makes a regression over this structured space is expected to generate correct new data. The interest in this architecture is that even if the distribution of the represented latent space is unknown (arbitrary since it is not imposed), the regression is possible in a controlled way since we have access parameters for the generation as a DNN input, a fact that is no always true in other AE architectures.

II.5 . Sobol' indices, δ-importance measure and SHAP results analysis as application of DNN metamodel

Based on the DNN metamodel validated in Subsection II.4.1, we applied it to the calculation of GSA indices and FI calculation for parameter ranking purposes, as introduced in Section II.3. We compute $25\mathrm{k}$ samples for the GSA calculation routine of the SALib Python-based sensitivity analysis library [125], where the twelve driving parameters have been varied by Latin hyper-cube sampling. That is, both Sobol' and δ-importance indexes were calculated efficiently in about 370 seconds, and the results are shown in Figure II.8 and Fig. II.9, respectively. In Fig. II.10, the FI ranking relies on SHAP values analysis, where the calculations were based on *KernelExplainer* within the Python-based library called SHAP [110].

From the cross-comparisons between the two GSA indices, one can notice that there is an overall high correlation between Sobol' and δ-importance indices regardless of the ECT components considered (i.e., absolute, real or imaginary). On the other hand, the comparisons between GSA and SHAP show overall good agreement in the order of indices with complete concordance in the real part. The biggest discordance between the analysis is in the index of the l_{C2} in the absolute value, being the more important

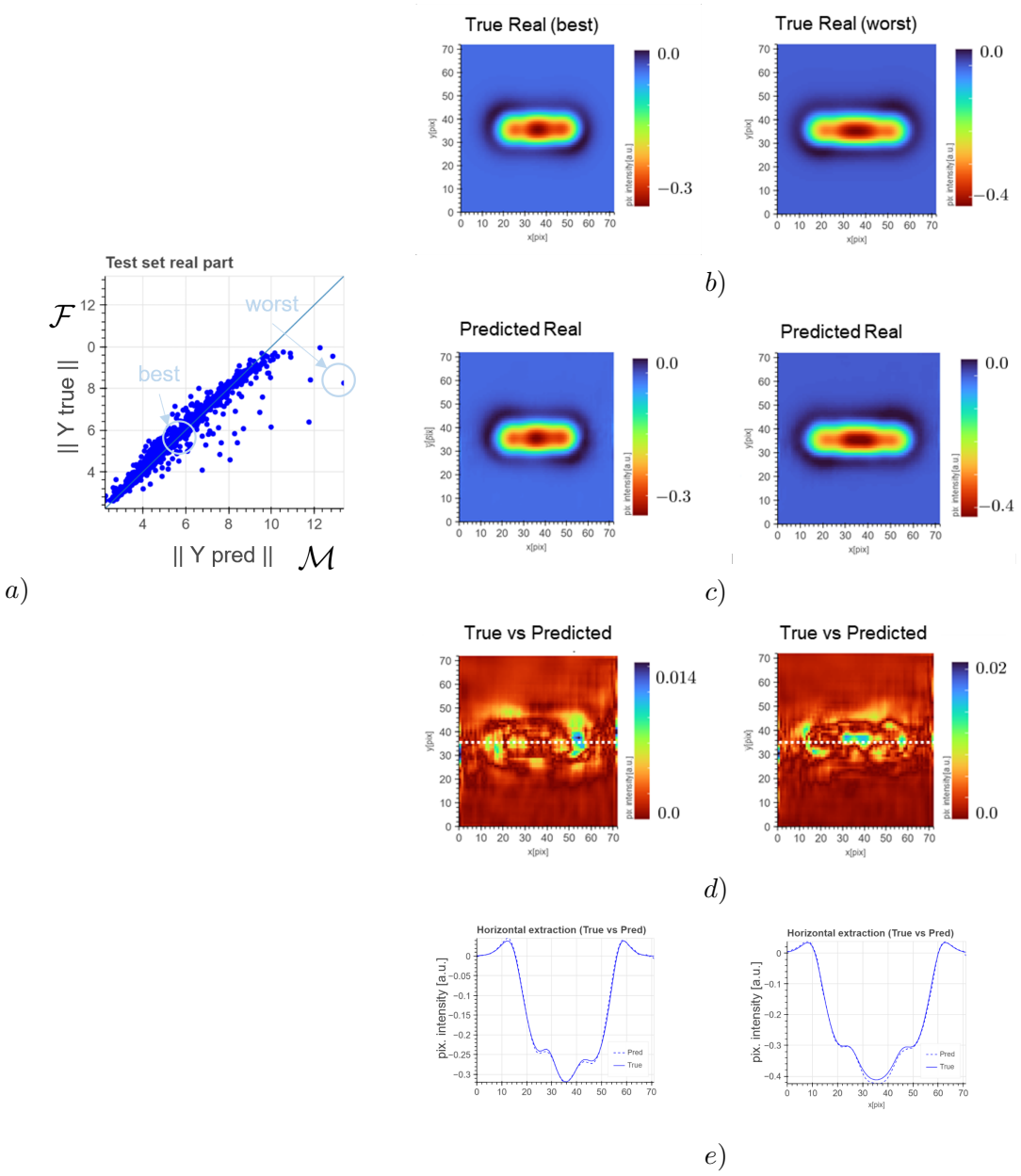


Figure II.4. Qualitative assessment of the DNN metamodel regressor for the test set prediction of the **real part** of ECT signals ($Re\{\Delta Z\}$). In a) the ground true versus predicted plot based on the Frobenius norm of the ECT C-scan. In b), c), d), e) the is two columns, the one at the left represent the one of the best prediction cases and the one at the right the worst prediction case, both based in the norm plotted in a). In b) is shown the true values, in c) the predictions, in b) the differences between the C-scans of GT and in e) predictions and the scan extractions along the horizontal cut sketched on d) plots.

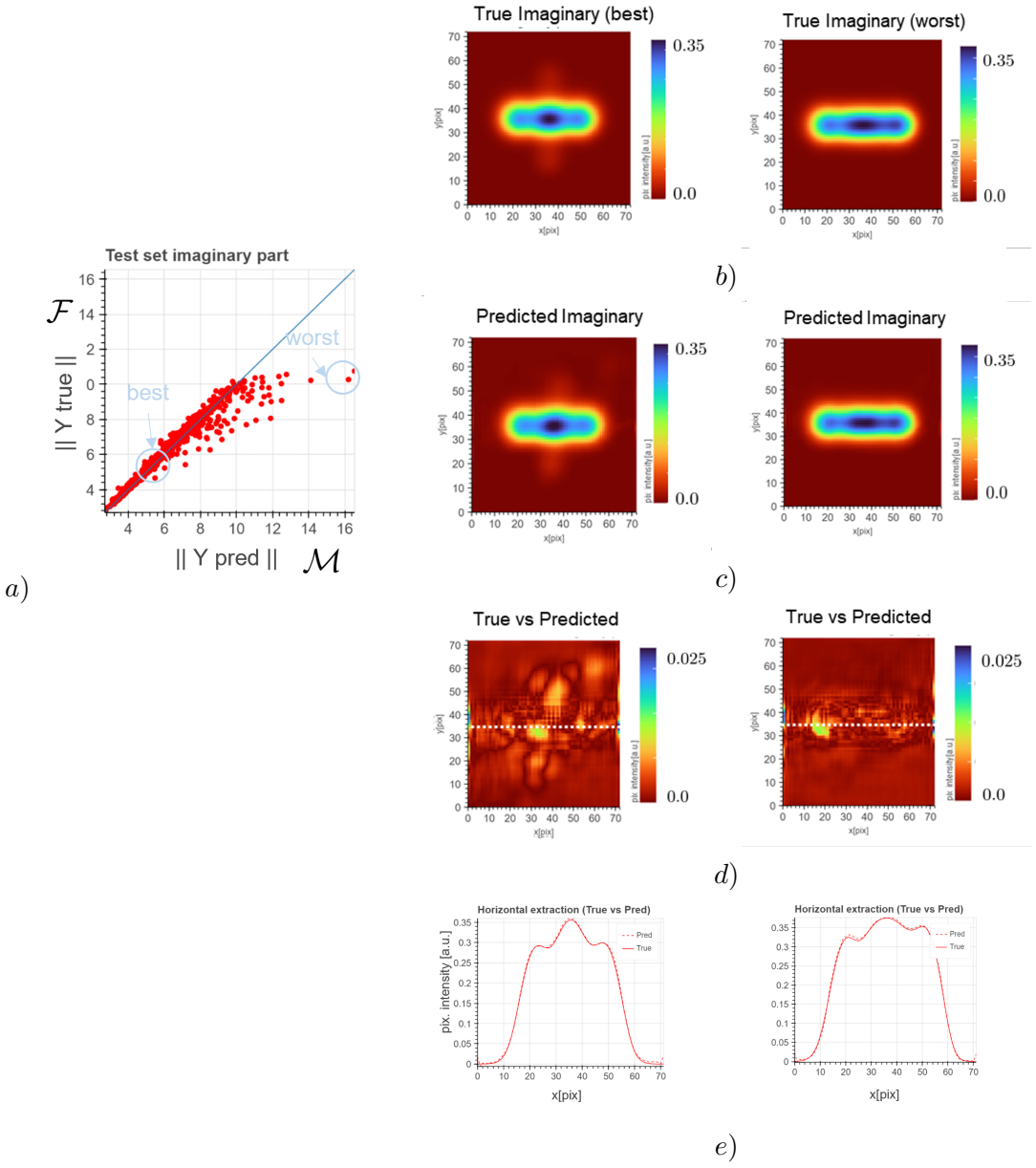


Figure II.5. Qualitative assessment of the DNN metamodel regressor for the test set prediction of the **imaginary part** of ECT signals ($Im\{\Delta Z\}$). In *a*) the ground true versus predicted plot based on the Frobenius norm of the ECT C-scan. In *b*), *c*), *d*), *e*) the is two columns, the one at the left represent the one of the best prediction cases and the one at the right the worst prediction case, both based in the norm plotted in *a*). In *b*) is shown the true values, in *c*) the predictions, in *b*) the differences between the C-scans of GT and in *e*) predictions and the scan extractions along the horizontal cut sketched on *d*) plots.

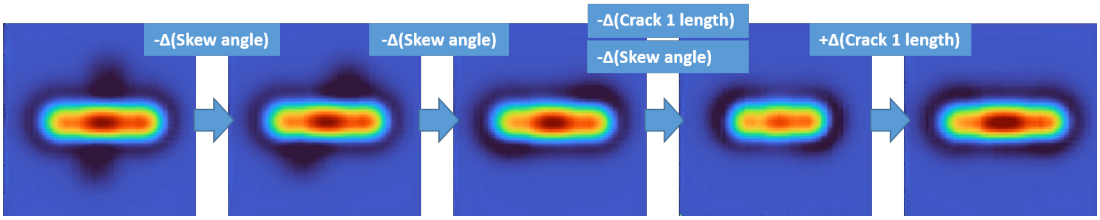


Figure II.6. Example of generation by DNN metamodel. The input parameters are varied by increasing or decreasing the skew angle and the upper crack length form an initial set of parameter that represents.

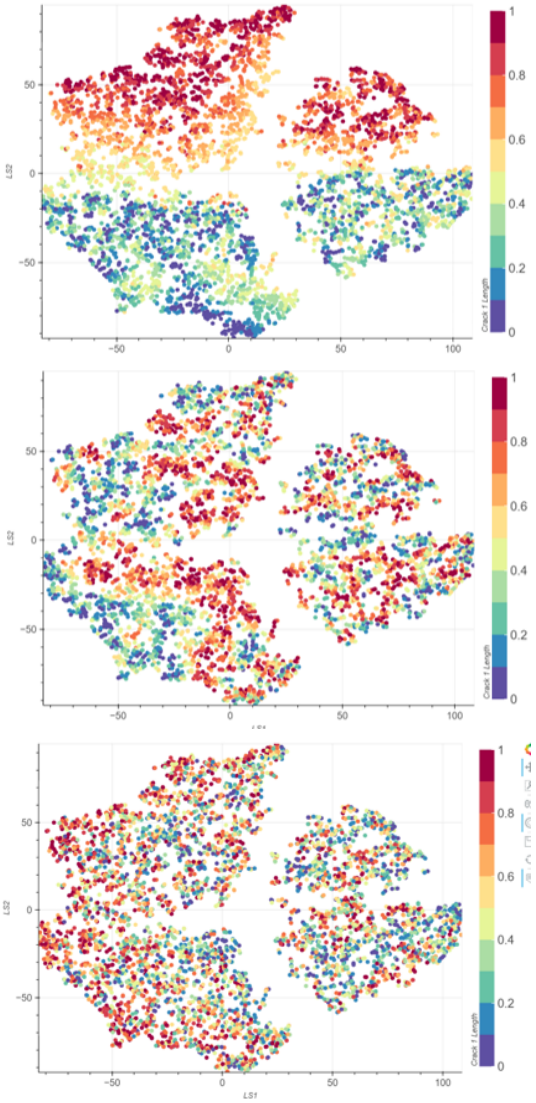


Figure II.7. DNN bottleneck layer representation based on $2D$-projection t-SNE projection. From top to bottom, points are colored accordingly to ϕ_{12}, l_{C1} and lo, respectively.

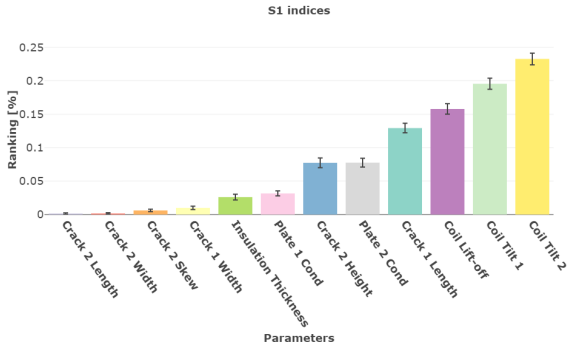
feature for SHAP while its Sobol' and δ-importance index is low. Such a particular behavior needs further investigation since it seems to be associated with the real part of the ECT signal (which is embedded into the absolute value, too). Unlike both the GSA methods, through SHAP values analysis, one may have deeper information on how the input parameters impact the variability of ECT signals collected by the coil. More specifically, thanks to the so-called "bee-swarm" plots, we can analyze the distribution of SHAP values for each of the twelve considered parameters as shown in Fig. II.11. Looking at Fig. II.11, one can notice that the coil position parameters (tilt and lift-off) and both the crack 1 parameters (namely the length and the width) have a symmetric impact. h_{C2} has a more asymmetric impact for high and low values. This behavior can be justified by considering the underlying physics. Indeed, the penetration of eddy currents into the medium diminishes exponentially accordingly to the skin depth.As a consequence, a very minor impact of crack 2 on the variation of coil impedance is expected for smaller crack heights compared to the larger one. Furthermore, the relation between the variation of coil impedance and h_{C2} is linear when crack 2 approaches the second plate upper surface, and it becomes non-linear when crack 2 breaks the second plate's upper surface. Based on these results, it is believed that the richer information provided by "bee-swarm" -like plots is a valuable tool to deeply analyze both GSA and FI results in many realistic inspection cases (not only linked to ECT inspection signals).

II.6 . Chapter outlook and perspectives

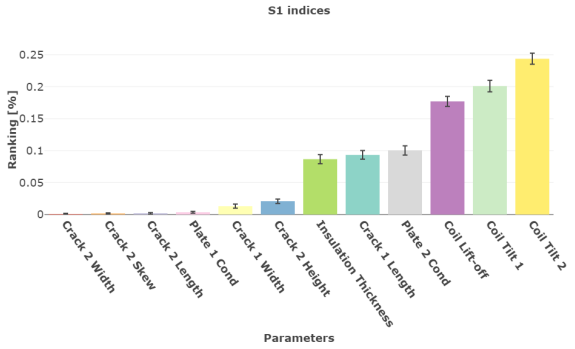
The present work proposes a tailored generative deep learning framework for very efficient simulation of ECT signals analysis. Our work shown how efficiently quantifying the sensitivity and establishing a feature importance ranking in parametric simulations that are often limited by the intrinsic computational burden associated with numerical simulations. In particular, we studied a fairly high cardinality problem (i.e., the size of input parameters). The present framework proposes a fast metamodel to overcome this limitation to perform GSA and FI computation.

A DNN regressor applied to an ECT inspection problem, where two arbitrarily oriented cracks are lying in a metallic multilayered planar structure, described by twelve parameters like specimen configuration, coil position, and crack geometries, has been studied. The DNN regression model has shown good prediction accuracy and high computational efficiency. Its application to global sensitivity and SHAP analysis has permitted to carry out of the complete study in a negligible amount of time if compared to the intensive use of the forward solver. Additionally, the analysis presented in this Chapter focuses on a comparative study between Sobol' indexes δ-importance measure and SHAP values. A satisfactory correlation of parameters ranking has been obtained regardless of the method considered. Furthermore, a deeper analysis of SHAP (i.e., bee-swarm plots) has permitted us to retrieve some physics-rooted behavior on the way that SHAP ranks the parameters. Furthermore, the analysis of inner-working mechanisms of the DNN architecture via the t-SNE manifold projection allowed to join the feature extraction procedure performed by the DNN model with the underlying physics linked to the use of a model-driven data set generation.

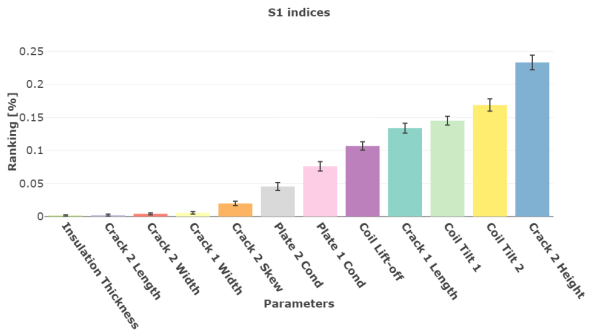
It is worth mentioning that the deep learning developed framework is not limited in terms of the problem cardinality and the problem addressed, as shown in next Chapter where a more complex data set is treated (indeed, the accuracy rather depends on the number of data available instead of the number of parameters to be considered). That is, more parameters can be considered by paying more time to train bigger DNNs with more data with a similar computational evaluation efficiency as a result. In the perspective of this work, we expect to explore the power of our AE backbone architecture for inverse problems, given that the structure of the latent space seems promising for this task.



a)

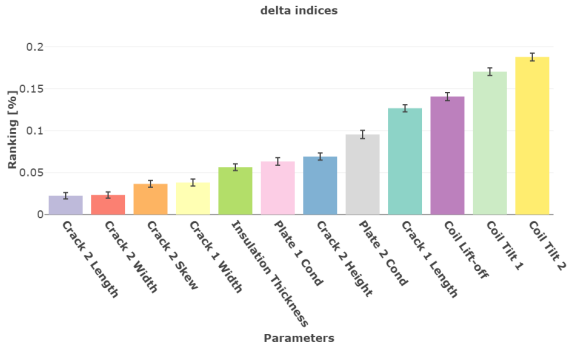


b)

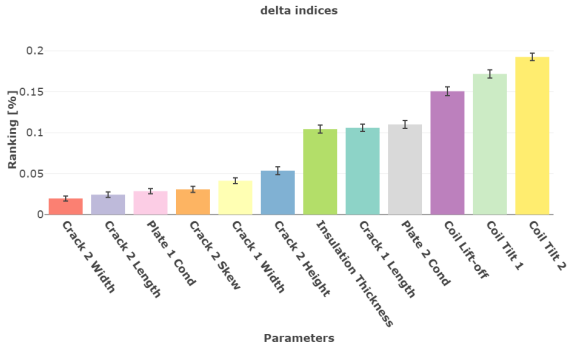


c)

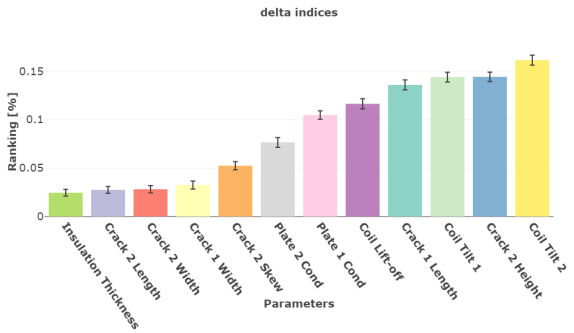
Figure II.8. Results of global sensitivity analysis through Sobol' first-order indexes. In (a), the results obtained from the extraction based on the $L2$-norm calculation performed on the absolute value of C-scan ECT signals. In $b)$ and $c)$ the results display the same calculation performed in $a)$, based on real and imaginary part of ΔZ, respectively.



a)

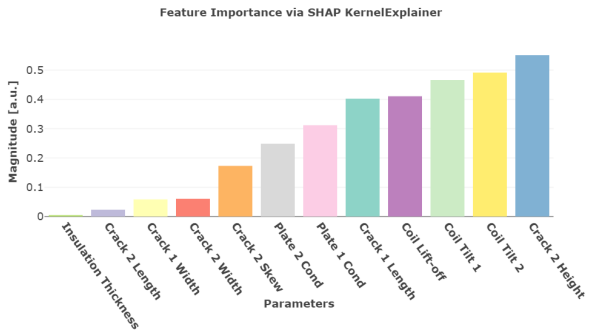


b)

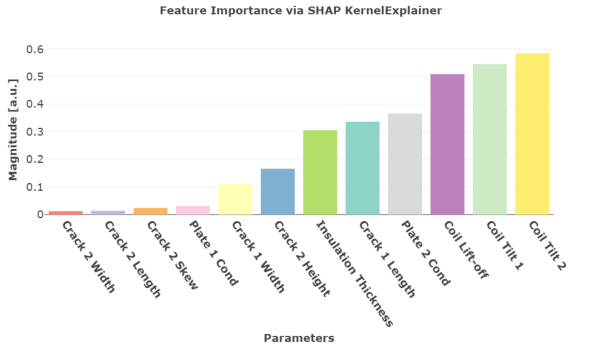


c)

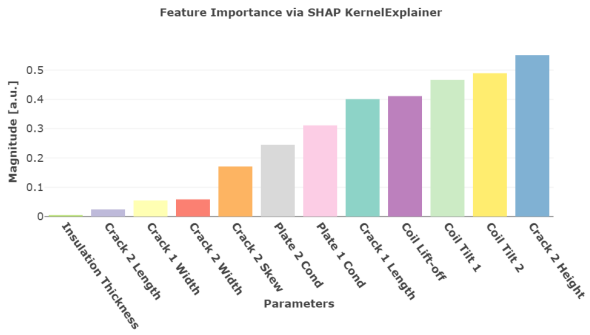
Figure II.9. Results of global sensitivity analysis through δ-importance index. In (a), the results obtained from the extraction based on the $L2$-norm calculation performed on the absolute value of C-scan ECT signals. In $b)$ and $c)$ the results display the same calculation performed in $a)$, based on real and imaginary part of ΔZ, respectively.



a)



b)



c)

Figure II.10. Results feature importance ranking based on SHAP values. In (a), the results obtained from the extraction based on the $L2$-norm calculation performed on the absolute value of C-scan ECT signals. In $b)$ and $c)$ the results display the same calculation performed in $a)$, based on real and imaginary part of ΔZ, respectively.

Figure II.11. Bees-warm SHAP plots values calculation applied to the DNN metamodel are given for the absolute (up-left), real (up-right) and imaginary (above) parts of ECT signals.

III - Development of a supervised generative approach in a TFM multi-fidelity data set to generate new robust data

About this section

In this chapter, we present the development of a *supervised* generative approach for a *multi-fidelity data set,* to generate more robust data than in the previous chapter. A new multi-fidelity image data set is produced by couples on two fidelity levels. The criteria respected during the production is to have a direct link between the two different fidelity levels. The first source is a simulation tool for an UT NDT&E application and the second is experimental data obtained from an equivalent mock-up. The differences between the fidelity level in the data set are studied. An adapted architecture is trained in a supervised way on the image data set. The labels are available in both fidelity levels, as well as the exact image mapping between the two fidelity levels. This a-priori information is used to supervise the training. The new generative data are evaluated to prove the potential for the application in an inverse problem. The limitations of the architecture are studied, particularly the need for a highly informed and labelled multi-fidelity data set.

III.1 . Introduction

Ultrasonic array imaging is among the most employed inspection methods for NDT&E. This technology enables fast scanning of industrial components; thus, wider inspection areas can be covered in a reasonable amount of time. In addition, ultrasonic arrays are rather flexible since they allow different acquisition modalities (that could be done in parallel), such as acquisitions with focused or steered plane waves or the so-called full matrix capture (FMC). The latter consists of recording inter-element signals corresponding to all transmitter-receiver pairs in the array.

The most common FMC data post-processing algorithm is the Total Focusing Method (TFM) imaging, which provides optimal focusing and spatial resolution throughout the region of interest if compared to other delay-and-sum methods based on focused beams, such as in [39]. Therefore, ultrasound array is widely adopted in complex inspection scenarios, such as in weld inspection, where mechanical properties, flaw topology, and geometrical profile represent challenging issues to be taken into account for analysing the acquisitions. Furthermore, the possibility to account for different reconstruction modes for TFM images, referred here as M-TFM version [126, 127, 128], has been proven to be even more powerful in detecting and characterising different types of defects appearing at different positions in welds. Indeed, the possibility of considering multiple wave paths allows it to effectively reach different locations within the weld where defects typically appear. M-TFM consists in exploiting different images of a given defect from the same set of FMC data, each corresponding to a particular wave path before and after interaction with the defect. In the context of crack-like defects imaging with quasi-vertical orientations, the reconstruction modes can be classified into two categories: those exploiting the diffraction echoes from the defect (e.g., direct, indirect modes) and those that exploit the specular echoes from the quasi-vertical face of the crack (e.g., half-skip, indirect modes) with ultrasonic paths including at least one reflection with the specimen interfaces, as well as the mode conversions (see Fig. III.1).

In practice, although the multi-modal approach can be useful to improve the inspection of welds, the quality of M-TFM images is strongly influenced by the actual knowledge of the velocities of longitudinal and transverse waves, the geometrical profile of the back-wall, as well as parameters of the experimental set-up (e.g., the position of the probe relative to the region of interest, water column height in the case of an immersion inspection). Uncertainties in such parameters can lead to errors in defect sizing and location, as well as imaging artefacts that make images difficult to interpret [129]. Furthermore, random noise sources coming from the acquisition and digital-to-analog conversion of signals (electronic noise) and from the wave scattering by heterogeneities in the material (structure noise) may impact the imaging.

The lack of knowledge about the aforementioned influential parameters can be partially mitigated by using adaptive TFM schema to deal with irregular geometries [130] or by employing optimisation strategies for anisotropic materials with uncertain elastic proprieties [131] with payback in terms of reconstruction efficiency. Due to the challenges mentioned above, the analysis of M-TFM images in complex inspection problems is not straightforward and might require numerical solvers with a non-negligible computational burden.

Very recently, simulation-driven inversion strategies have been applied with success in NDT&E, mixing ML, and UT [9, 1]. Those strategies based on time-domain signals or imaging data have also been studied in the context of structural health monitoring [2, 3, 5, 18, 8, 132, 6, 7], as well as in other domains such as eddy current testing [133, 134, 135]. The possibility to handle both numerical and experimental data and combine them in a tailored learning algorithm increases the efficiency of the deep learning (DL) algorithms, conceived to detect and size defects [45, 4, 17, 20]. In [136], the authors proposed a DL encoder-decoder architecture dealing with FMC data in order to suppress artefacts in reconstructed TFM images automatically. Once FMC experimental data are provided as inputs to the encoder-decoder, the DL architecture automatically performs denoising, reducing the presence of artefacts in TFM images.

DL has been adopted for data-augmentation strategies in [137]. It is worth noting that for the problems mentioned above, even the most accurate numerical solvers cannot fully reproduce some patterns that appear in experimental M-TFM images in a suitable computational time (i.e., a too-large combination of factors should be considered). This is often due to uncertainties or the lack of knowledge of input parameters.

The present study aims at developing a fully-controlled ML generative model targeting high-dimensional UT inspection problems embedding uncertainties coming from the lack of knowledge on inspection parameters and noise sources coming from unknown factors (e.g., electronic or structural noise, etc.). To this end, we propose a supervised learning schema based on physics-driven data issued from simulation tools aiming at replacing forward solvers in the massive and controlled generation of data used in advanced and time-consuming studies. In our analysis, we showed how a tailored conditional U-Net (cU-Net) architecture enables high-quality multi-fidelity M-TFM reconstructions based on both numerical and experimental data. Moreover, we provide an analysis of the architecture's inner working structure by showing how the regression procedure is performed.

This Chapter is structured as follows: Section III.2.1 exposes the principle of M-TFM used to produce the high- and low- fidelity data set. Section III.2 describes the architecture and the basis of the inner layers. In the same section, the loss function and the performance evaluations are summarised. The section III.3 shows the implementation of this framework on the M-TFM data. A description of data production is given, along with the hyper-parameters chosen for the architecture in this section. Additionally, an exploration of the inner activation of the trained Deep Neural Network (DNN) is commented. The conclusions are summarised in Section III.4. A discussion on the perspectives and possible applications are exposed in III.5. We have made the code publicly available, along with a pre-trained network[1]. An accompanying video can be found under the same link.

III.2 . Supervised approach in a TFM multi-fidelity data set

In this Chapter, we propose to tackle the problem of the generation of high-quality M-TFM images by employing a surrogate model that accounts for simulations and experimental measurements together. To this end, a supervised learning strategy relying on the use of a DNN architecture has been developed. More precisely, an end-to-end cU-Net such as those presented in [138, 139] has been conceived. The DNN is designed to account for both numerical and experimental M-TFM images. The cU-Net architecture is trained with both simulations as low-fidelity data source and experimental measurements

[1] https://github.com/geragranados/M_TFM_cUNet

as high-fidelity data source, along with physical parameters associated with the numerical set-up considered (e.g., flaw(s) size and orientation, specimen geometry, phase velocities of elastic waves, M-TFM reconstruction parameters, etc.) that plays the role of conditioning variables. Furthermore, to properly address the complexity of the targeted problem, the proposed cU-Net exploits advanced deep learning constitutive blocks such as the Feature-wise Linear Modulation (FiLM), proposed by [116, 140], and a modified version of the spatial transformer block [117], named in this work as parametric Spatial Transformer (pST). Our cU-Net model can be considered as a multi-fidelity model [99] aiming at enhancing the numerical simulation capabilities in terms of both accuracy and efficiency thanks to an almost real-time M-TFM image generation.

It is worth pointing out that, in contrast to other DL and ML close-box models based on a purely data-driven approach, the multi-fidelity model developed in this work can be considered to be closer to an open-box since the physics-based knowledge is injected into the learning procedure via simulations. The objective of this strategy is to teach the surrogate model to encode all the useful information from the simulated M-TFM images while generating images close to the experience. At the same time, the encoder is intended to learn how its latent features are influenced by the simulation parameters to be able to rapidly reproduce several M-TFM images spanning the space of parameters.

III.2.1 . TFM multi-fidelity data set

This section first recalls the principle of M-TFM imaging and its application to the inspection of welds where crack-like defects may appear and propagate along the chamfer. In addition, we describe the reconstruction modes that will be exploited to evaluate the constraint generative model procedure, and we provide some insight into the impact of reconstruction uncertainties on the M-TFM images.

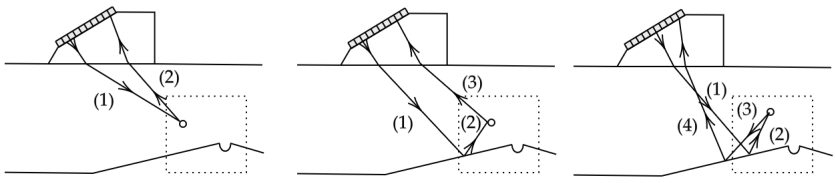
The multi-modal imaging use-case considered in this chapter is shown in Fig. III.1. It is a common NDT&E configuration for butt welds where an ultrasonic array is attached to a Rexolite wedge to perform an oblique inspection with either incident longitudinal (L) or transverse (T) waves depending on the wedge angle. The complex geometry of the steel mock-up is representative of a butt V-shaped weld with tilted interfaces on both sides of the weld root. A potential critical defect that can occur in welded parts is a crack that may propagate near or along the weld chamfer, and it is represented by a notch in the mock-up. A Region of Interest (ROI) for the Eq. I.9 is defined in the surroundings of the defect. A major challenge in NDE is the detection and characterisation of such defects as early as possible before their growth threatens the structural integrity of the component. It is precisely for this purpose that TFM has been extended to multi-modal imaging.

Multi-modal TFM is the formation of several images of the same defect from a single FMC data set with appropriate modes, and these images can be combined to obtain a more realistic representation of the defect [128, 141, 142, 143]. These views aim to utilise ray paths that maximise the viewing angle of any particular defect, making visible defect responses across these views more likely.

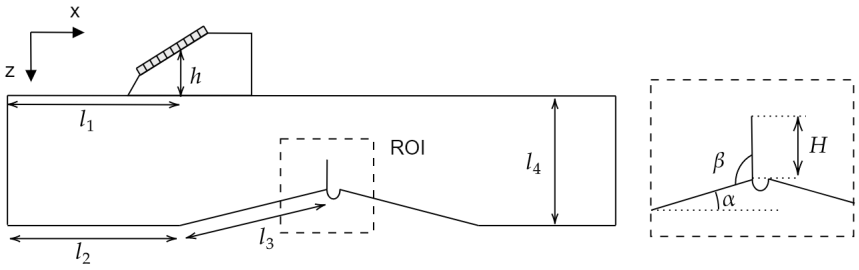
When no reflection on the back-wall occurs along the path between the elements and the running point P (Fig. III.1(a)), the mode is called direct mode, in contrast with the half-skip and indirect modes, characterised by one or multiple reflections from the upper and lower surfaces. The combination of longitudinal (L) and transverse (T) waves leads to four direct reconstruction modes. L-L (or T-T) designates the mode for which the return trip is with longitudinal wave (or transverse wave), while L-T and T-L designate the modes with conversion when the incident wave-field interacts with the defect [126]. Additionally, half-skip modes denoted $P_1P_2-P_3$ and indirect modes denoted $P_1P_2-P_3P_4$ can be considered, where P_i = L o T (i.e., see III.1(a)).

The subset of M-TFM images under the selected modes (direct, half-skip, and indirect) used in this work are shown in Fig. III.2. The inspection is done in a complex specimen such as the one shown in Fig. III.1(b), where an artificial slot is considered a root-like crack.

In this work, the numerical simulation of the UT array inspection represents the low-fidelity data. The simulation data are produced on a parametrical semi-analitical modelling on CIVA (Eq. I.6, I.7, I.8). Meanwhile, the experimental mock-up acquisitions represent high-fidelity data. In both cases,



(a)



(b)

Figure III.1. (a) Ray paths are associated with three families of reconstruction modes: direct modes including two sub-paths; half-skip modes with three sub-paths; and indirect modes with four sub-paths. Each ultrasonic sub-path corresponds to the propagation of L or T waves. (b) Contact inspection configuration with a Rexolite wedge for imaging a back-wall breaking notch machined in a steel mock-up representative of a butt weld.

the M-TFM imaging technique is applied to the temporal data to produce the image data set (Eq. I.9). As a result, a multi-fidelity data set of M-TFM images is then defined as $\mathcal{D} = \{(\mathbf{X}_n, \mathbf{p}_n, \mathbf{Y}_n) : n \in \{1, ..., M\}\}$, where M is the number of samples (Eq. III.1). An instance $(\mathbf{X}_i, \mathbf{p}_i, \mathbf{Y}_i)$ represents a low fidelity image $\mathbf{X}_i$ with labels $\mathbf{p}_i$ (simulation parameters) and its high fidelity couple $\mathbf{Y}_i$, respectively. Couples of images are created by respecting the same parameters for both fidelity levels. The data set described is then separated into training, validation, and test set. $\mathcal{D}$ contains:

$$\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{W \times H \times C}, \quad \mathbf{p} \in \mathcal{P} \subset \mathbb{R}^{n_p};$$

$$\mathbf{Y} \in \mathcal{Y} \subset \mathbb{R}^{W \times H \times C}, \tag{III.1}$$

where W, H, C are the weight, height and number of channels for the images in both fidelity levels, and n_p size of the vector of parameters associated to each couple of images $\mathbf{X}_i$ and $\mathbf{Y}_i$.

It is worth noticing that $\mathbf{X}$ and $\mathbf{Y}$ have to be intended as two fidelity data sets. Indeed, $\mathbf{X}$ represents a signal in the data set like $\mathbf{Y}$, but it belongs to a different fidelity level. This setting differs from the previous chapter where just one fidelity was explored. The next section aims to specifically introduce the deep neural network details and choices made to this learning framework.

III.2.2 . Conditional U-Net architecture

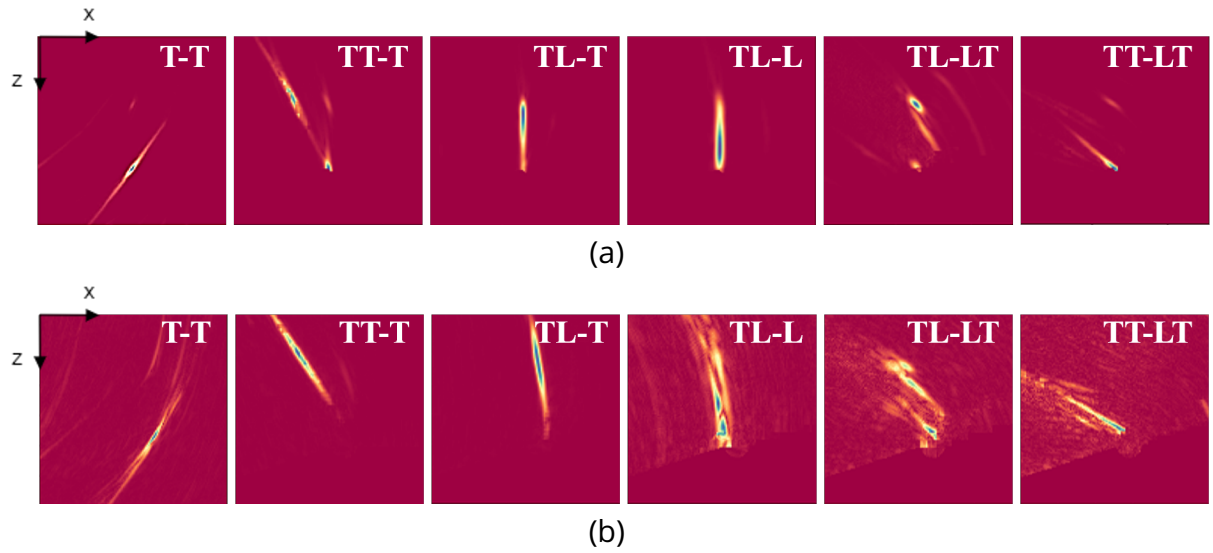


Figure III.2. Examples of experimental M-TFM images with set-up are shown in Fig. III.1. M-TFM images of the back-wall breaking notch are given for direct-, half-skip, and indirect mode reconstruction with (a) the actual specimen given a celerity and flaw geometry and (b) the same specimen with uncertainties in celerity T (cT) and α. Upper row: $cT = 3230$ m/s and $\alpha = 14°$. Lower row: $cT = 3380$ m/s and $\alpha = 18°$. The images are show in a normalised linear scale.

The conditional U-Net (cU-Net) is a modified architecture from the original U-Net [144]. The classical U-Net is essentially an encoder-decoder structure with skip-connections across the encoding and decoding trunks. Our cU-Net (Fig. III.3) has as input a simulated image and its labels. The architecture is intended to be used as a parametric surrogate model and a realistic data generator, since the input is expected to be an experimental image. The objective is to learn the link between the simulation parameters (labels) $\mathbf{p}$ and the latent features $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}$ associated to the input image $\mathbf{X}$, although learning a disentangled latent manifold still represents a major scientific challenge [145]. In this sense, the trained cU-Net will output new realistic samples $\hat{\mathbf{Y}}$ from the one simulated image $\mathbf{X}$, based on the variation of $\mathbf{p}$ (considered as input labels).

This architecture was conceived to learn from a multi-class data set, such as a database of M-TFM images for different mode reconstruction or different types of flaw geometry. The cU-Net is fed with batches of data $(\mathbf{X}, \mathbf{p})$ of size B. The model is expected to map $(\mathbf{X}_i, \mathbf{p}_i)$ to $\mathbf{Y}_i$. The first part of the network $F : \mathcal{X} \to \mathcal{Z}$ (yellow blocks in Fig. III.3) encodes the images $\mathbf{X}$ into a conditioned latent space representation $\mathbf{z}$ (features map of green block in Fig. III.3). This encoding is conditioned by $\mathbf{p}$.

The decoder or generator G (blue blocks in Fig. III.3) takes the space $\mathcal{Z}$ and the skip-connections to compute the images $\mathbf{Y} \in \mathcal{Y}$. Additional convolution layers (the so-called *synthesis* layers, depicted in violet in Fig. III.3) are stacked on top of the generator G to improve the reconstruction. In this study, both F and G are featured by a stack of convolutional layers, with parametric Rectified Linear Unit (pReLU) activation functions [115], except for the synthesis block layer. pReLU set the LeakyReLU activation coefficient as a learn-able parameter for the DNN.

In order to inform the latent manifold with the M-TFM parameters $\mathbf{p}$, a FiLM-pST block is inserted within each convolutional layer in the encoder F. The objective of these operators is to structure the latent representation at the end of the encoder by learning the relation of parameters (labels) and features in different levels of resolution. The encoder extracts a sequence of features at each layer from a given input $\mathbf{X}_i$. we denote the output of a layer k as $\mathbf{X}_k \in \mathbb{R}^{r_k \times r_k \times C_k}$. r_k is the resolution at k after the down-sampling from $k-1$ and C_k is the number of filters (or channels) for k. At the FiLM and pST layers, the $\mathbf{X}_{k+1}$ has the same filter resolution and number of channels as the input $\mathbf{X}_k$.

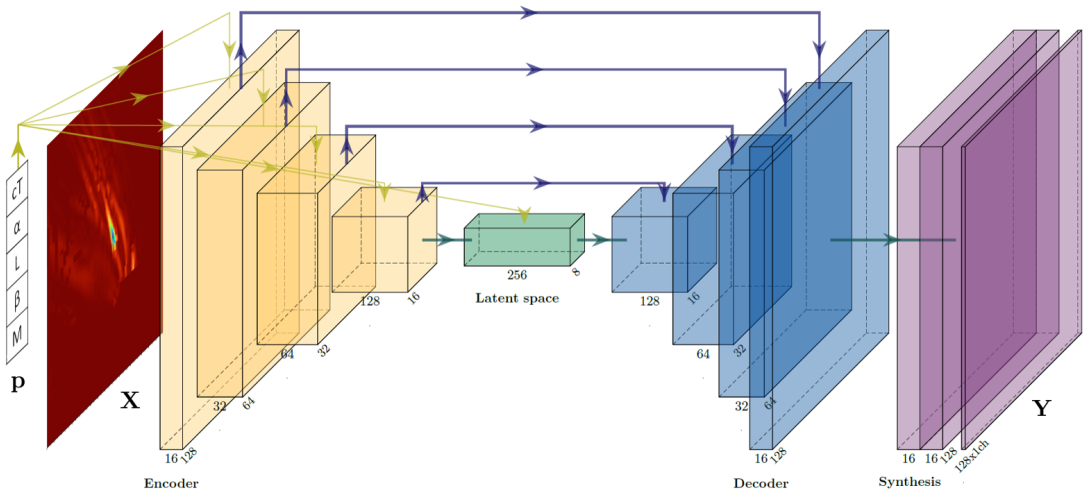


Figure III.3. Schematic representation of cU-Net architecture, created with [146]. The input is a M-TFM image and its labels (simulation parameters). The DNN's encoder is represented in yellow, the latent space representation in green, and the decoder in blue. The violet layers represent the synthesis block. U-net skip-connections are shown by blue lines, and labels forward propagation on the encoder are represented by yellow arrows. An example of the number of filters and resolution at each block is given.

A FiLM layer learns new per-channel statistics by applying the scales $\boldsymbol{\gamma}_k$ and the bias $\boldsymbol{\beta}_k$ to a normalised representation of $\mathbf{X}_k$ (Fig. III.4(a)). The values for $\boldsymbol{\gamma}_k$ and $\boldsymbol{\beta}_k$ are learn-able functions of $\mathbf{p}$. FiLM modifies the relative importance of features for the subsequent convolution operation $k+1$. The inference from the labels $\mathbf{p}$ is performed by a dense layer. Eq. (III.2) represents the operation done per-channel (C_k) in the encoder, where $\mathbf{p}$ is the image label, $\boldsymbol{\gamma}_k(\mathbf{p})$ and $\boldsymbol{\beta}_k(\mathbf{p})$ are two outputs of the dense layer. FiLM$(\mathbf{X}_k, \mathbf{p})$ is the output of this layer, so the modified input features. In this layer, the mean and variance are computed per batch to normalise $\mathbf{X}_k$. ϵ is added for numerical stability and precision.

$$\text{FiLM}(\mathbf{X}_k, \boldsymbol{p}) = \boldsymbol{\gamma}_k(\boldsymbol{p}) \cdot \frac{\mathbf{X}_k - \mathbb{E}[\mathbf{X}_k]}{\sqrt{\sigma^2[\mathbf{X}_k] + \epsilon}} + \boldsymbol{\beta}_k(\mathbf{p}), \tag{III.2}$$

where $\boldsymbol{\beta}_k : \mathbb{R}^{C_k}$ and $\boldsymbol{\gamma}_k : \mathbb{R}^{C_k}$. $\mathbb{E}[\cdot]$ and $\sigma^2[\cdot]$ represent the empirical average and variance respectively. ϵ is a small value used for numerical stability, normally set to 1×10^{-3}.

Most of CNNs apply a normalisation before each activation. The FiLM module replaces the normalisation layer (e.g., IN layer) in the encoding stream (F), giving the DNN a stable convergence and the degree of freedom to use the input $\mathbf{p}$ to improve the reconstruction and to allow the regression on the label space, as is shown in section III.3.

Upon each FiLM layer in the encoder F, the proposed architecture stacked a pST layer. The original ST layer proposed by [117] was herein modified to make use of the parameter $\mathbf{p}$. An pST layer applies a spatial transformation capable of rotating, translating, and scaling a 2D input feature map $\mathbf{X}_k$ (Eq. (III.3) and Eq. (III.4)), via the per-instance matrix $\mathcal{T}$ affine transformation

$$\text{pST}(\mathbf{p}, \mathbf{X}_k) = \mathcal{T}(\mathbf{p}) \circ \mathbf{X}_k, \tag{III.3}$$

$\mathcal{T}$ applies the transformation by computing $T_{\boldsymbol{\phi}}$ where $\boldsymbol{\phi} : \mathbb{R}^6$ and $\mathcal{T}_{\boldsymbol{\phi}} : \mathbb{R}^{3\times3}$, that reads:

$$\boldsymbol{\phi}(\mathbf{p}) = [\phi_1(\mathbf{p}),\ \phi_2(\mathbf{p}),\ \phi_3(\mathbf{p}),\ \phi_4(\mathbf{p}),\ \phi_5(\mathbf{p}),\ \phi_6(\mathbf{p})],$$

$$T_{\boldsymbol{\phi}}(\mathbf{p}) = \begin{bmatrix} \phi_1(\mathbf{p}) & \phi_2(\mathbf{p}) & \phi_3(\mathbf{p}) \\ \phi_4(\mathbf{p}) & \phi_5(\mathbf{p}) & \phi_6(\mathbf{p}) \\ 0 & 0 & 1 \end{bmatrix}. \tag{III.4}$$

Analogously, as in the FiLM layer, pST layer makes use of an dense layer (called localisation net) that infer $\boldsymbol{\phi} = \boldsymbol{f}_\theta(\mathbf{p})$ from $\mathbf{p}$ (Fig. III.4(a)). The pST layer takes the input vector of parameters $\mathbf{p}$ to infer the spatial transformation for all channels. This transformation is applied to the output of the previous layer.

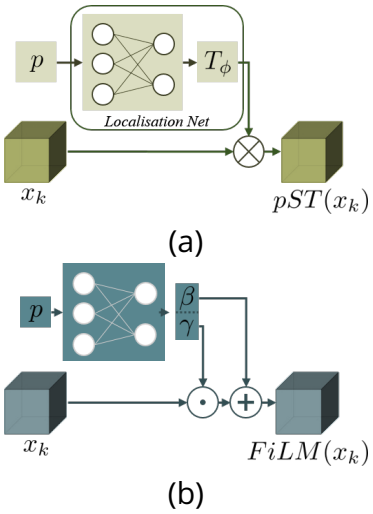


Figure III.4. Schema of the parametric (a) FiLM and (b) pST layers. FiLM inner normalisation is not represented here for clarity.

On the other hand, the generator G makes use of Instance Normalization (IN) [118] between each convolutional layer. IN showed impressive performance for feed forward stylising [147]. The DNN implements the pixel-wise L2-norm and the Focal Frequency Loss (FFL) [148] (Eq. (III.5)) as image reconstruction loss. The reconstruction loss $\mathcal{L}_{rec}$ reads:

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}, \mathbf{p}, \boldsymbol{\theta}) = {}^1\!/_B \sum_{i=1}^{B} [||\mathbf{Y} - u_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{p})||_2 + FFL(\mathbf{Y} - u_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{p}))], \tag{III.5}$$

where $u_{\boldsymbol{\theta}}$ is the cU-Net and $\boldsymbol{\theta}$ its learn-able parameters and B the size of the batch.

The addition of the described modules is justified by the feature evolution observed on the images in $\mathcal{X}$ when exploring $\mathcal{P}$. For instance, a couple $(\mathbf{X}_i, \mathbf{p}_i)$ and a couple $(\mathbf{X}_j, \mathbf{p}_j)$ in the same reconstruction mode (e.g., T-T) are not far from each other in $\mathcal{X}$ if $\mathbf{p}_i$ and $\mathbf{p}_j$ are closed enough. Scaling, rotations, and translations in some features, together with changes in the echo shape, can be observed in $\mathbf{X}_j$ with respect to $\mathbf{X}_i$.

The transformations needed to obtain $\mathbf{X}_j$ from $\mathbf{X}_i$ are learnt by the module FiLM-pST in F. Therefore, the DNN is expected to set $F(\mathbf{X}_i)$ and $F(\mathbf{X}_j)$ near on the $\mathcal{Z}$ space. As a result, $u_{\boldsymbol{\theta}}$ serves as a parametric surrogate model. The DNN is expected to map the $\mathcal{P}$ space into the image feature space $\mathcal{X}$ and $\mathcal{Y}$ (injective mapping). For instance, if $\mathbf{X}_i$ and $\mathbf{X}_j$ are two samples from the training set and for the test set, respectively; the couple $(\mathbf{X}_i, \mathbf{p}_j)$ must generate the instance $\mathbf{Y}_j$ (with $\mathbf{p}_i \neq \mathbf{p}_j$), without the need to know $\mathbf{X}_j$ during the training. The model does not implement any additional loss for this task besides $\mathcal{L}_{rec}$, so the $\mathcal{P}$ to $\mathcal{Y}$ conditional mapping is learnt thanks to two intrinsic characteristics of the DNN. The first characteristic is the FiLM-pST module introduced in the encoder F, which learns to condition the features extracted from the input image with different labels. The second one is related to the supervised

framework adopted. That is, $\mathbf{X}_i$ and $\mathbf{Y}_j$ have the same label $\mathbf{p}_i$ associated during the training since the data set is presented by a couple of instances from both fidelity levels. As a consequence, the output image $\hat{\mathbf{Y}}_j$ can be directly labelled by input parameters $\mathbf{p}_j$. The results of this application are shown later in section III.3.

On the other hand, $u_{\boldsymbol{\theta}}$ maps the space $\mathcal{X}$ into $\mathcal{Y}$ by learning the differences between the images from $\mathcal{X}$ and from $\mathcal{Y}$. Those differences can be appreciated in the Signal-to-Noise Ratio (SNR) variation, the flaw echo location and shape. The encoder-decoder couple acts here as a realistic data generator. guided by the simulation input.

III.2.3 . Conditional surrogate model validation

Some metrics are implemented to measure the echo localisation error and the pixel intensity error for the couples $\mathbf{Y}$ versus $\hat{\mathbf{Y}}$. The metrics express the reconstruction quality for the DNN either for the task of realistic generation data or the parametric model application. The maximum value of the L1-norm for an instance expressed by Eq. (III.6) is used to quantify the error at the peak value, normally located in the echo region of the M-TFM image. Another metric is the Mean Absolute Error (MAE) which quantifies in average the reconstruction error for $\hat{\mathbf{Y}}$. The position error of the maximum obtained by Eq. (III.6) is also a metric for evaluating the performance on the test set.

$$\mathcal{E}_{rec}(\mathbf{X}_j, \mathbf{p}_j, \mathbf{Y}_j) = |\mathbf{Y}_j - u_{\boldsymbol{\theta}}(\mathbf{X}_j, \mathbf{p}_j)|_1 \tag{III.6}$$

III.3 . Numerical validation

The present section illustrates the outcome of the trained architecture on the test set, in other words, images never seen by the DNN. Firstly, a realistic image generation is evaluated for the test set through the adapted metrics. Secondly, a conditional generation through regression in the input $\mathbf{p}$ is evaluated and compared to the realistic generation error. Lately, an exploration by a 2D projection of the output features of the latent space block is used to show the data generation potential. The shown structure of those features tends to explain how the DNN is capable of acting as a parametric surrogate model to generate new M-TFM instances. With the same objective of explaining how the DNN works, the exploration of the features extracted by the FiLM layer and the pST layer are shown to interpret their role in the architecture.

The presented architecture was tested in a M-TFM image data set with two fidelity levels: simulation and experimental acquisitions. The data was obtained by a parametric simulation by sampling the $\mathcal{P}$ space as shown in Table III.1. The mock-up in Fig. III.1 follows the configuration given by [142], with a geometry described by $l_1 = 50\,\mathrm{mm}$; $l_2 = 60.5\,\mathrm{mm}$; $l_3 = 50\,\mathrm{mm}$; and $l_4 = 42\,\mathrm{mm}$, and a ROI of a square of $30\,\mathrm{mm}$ of length. Consequently, the data set $\mathcal{D}$ of $M \sim 6\mathrm{k}$ single-channel images is produced. Half of the images belong to simulated instances, while the second half is the experimental couples. The data set is separated in the proportions of 85%, 12%, 3% for the training ($\mathcal{T}$), validation ($\mathcal{V}$) and test set ($\mathcal{S}$), respectively. A random selection of the group $(\mathbf{X}_i, \mathbf{p}_i, \mathbf{Y}_i)$ is done to generate disjoint sets so $\mathcal{D} = \mathcal{T} \cup \mathcal{V} \cup \mathcal{S}$ with $\mathcal{T} \cap \mathcal{V} \cap \mathcal{S} = \emptyset$. The image size is (W, H, C) = (128 pixels, 128 pixels, 1 channels). The input label size is $d = 5$, containing the celerity of the transversal ultrasonic wave (c_T), the back-wall angle (α), the reconstruction mode (M), and the flaw geometry expressed by the height (L) and the tilt (β). Those labels are re-scaled to be expressed in the range of $[0, 1]$ (Fig. III.5). For those labels that had semantic representation (e.g., reconstruction modes), a simple dictionary is created to map the names to numerical values in the same range as other numerical labels.

The number of channels per block is augmented by a power of 2 in the encoder F, starting with 16 channels after the input to ending up with 128 channels before the latent space block. The resolution of the learnt filters at the end of every block of F is decreased by the down-sampling operation. The resulting resolutions by block at the encoder are 128, 64, 32, and 16. Each resolution block contains two FiLM-pST blocks; the first is sandwiched between a convolutional layer and its pReLU activation. The activation is followed by the second FiLM-pST block that is set between the second convolution

Table III.1. Parameter space definition for M-TFM simulation and experimental data set. Vertical: $\beta - \alpha = 90°$; Tilted: $\beta - \alpha = 74°$

Parameter space (labels)	Range of values	n-points in $\mathcal{P}$
Flaw tilt (β)	Vertical ; Tilted	2
Flaw size (L)	$3\,\text{mm}; 10\,\text{mm}$	2
T wave celerity (c_T)	$[3080; 3380]$	9
Geometry configuration angle (α)	$[10; 18]$	9
Modes used in reconstruction (M)	T-T;TT-T;TT-L;TL-T TL-L;TT-TT;TL-LT; TT-LT;ALL	9
	Total of instances: (couples)	2916 (324 per mode)

operation of the block and its pReLU activation. This layer sequence ends up with a down-sampling operation before passing to the next resolution block.

Regarding the latent space block, the structure is similar to an encoder block since it accounts for the FiLM-pST layer. The difference is that there is no down-sampling. As a consequence, the block is built as follows: FiLM-pST + pReLU + convolution layer + FiLM-pST+ pReLU. The resolution at this level is 8x8 with 128 channels for all the layers. The objective of this arrangement is conditioning the encoding of the DNN throughout all layers until the first up-sampling in the decoder so the labels build a structured latent space for the generator's input.

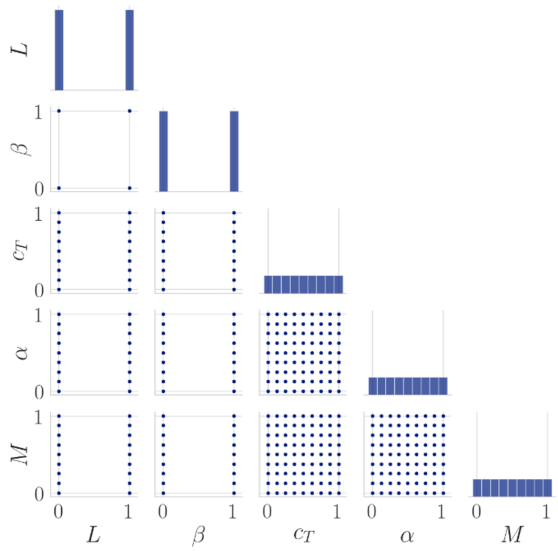
For the decoder G, we inverted the order in terms of resolution and number of channels, keeping the same number of layers as the encoder. Instead of the down-sampling, the up-sampling operation at the end of each resolution block is used to increase the resolution. The feature normalisation is done by the IN layers before each activation layer on the decoder. Finally, the synthesis block has three convolutional layers with the same resolution as the expected output but with more channels in the first two convolution layers: 16 channels. This block also implements the IN, but it differs from the encoder's blocks since a hyperbolic tangent activation function is used on it. The activation is choice to avoid saturation around the noise range that is near to zero on the normalized TFM images.

III.3.1 . Analysis of the training phase: the role of FiLM-pST

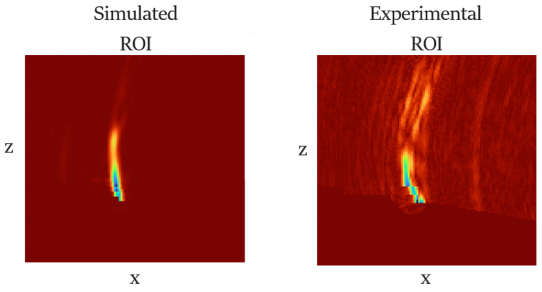
This section shows the convergence of the DNN driven by its architecture, particularly by the presence of feature operators such as the FiLM and pST layers. The evolution of loss versus the epochs is shown in Fig. III.6 for the validation and test. The batch size (B) is chosen equal to 128, and an Adam optimiser with a learning rate of 0.001 is chosen. The training time is $\sim$ 3h on an NVIDIA Quadro RTX 6000 GPU. The early stopping activated at around 417 epochs. The trained model is then applied to the test examples in the following sections.

During the training, we observed that adding the parameter of reconstruction modes (M) to the label vector helped the reconstruction quality and the convergence speed, even if a regression over these parameters does not have any physical sense (e.g., a hybrid T-T+TT-L mode). This parameter plays a role in the structure of the latent space and helps to the DNN convergence. In the opposite sense, the simple training of the DNN without any conditioning (no FiLM-pST blocks, no $\mathbf{p}$ in the input, so an U-Net architecture) gets the best reconstruction for $\hat{\mathbf{Y}}$. Consequently, the conditioning worsens the reconstruction capability, but it introduces the surrogate modelling potential into the architecture: generation of M-TFM conditioned by physical parameters.

During the test phase, the $\mathcal{S}$ set is used to evaluate the DNN performance in order to show the capability of the cU-Net as a multi-fidelity generator. The couples $(\mathbf{X}_j, \mathbf{p}_j) \in \mathcal{S}$ are forwarded from



a)



b)

Figure III.5. The M-TFM data set contains a set of images labelled by the simulation parameters and classes. (a) Sampling of parameters $\mathbf{p} \in \mathcal{P}$ (simulation parameters for each coupled of M-TFM images). The blocks represent the distribution of the sampling for each parameter and the dots gives a quick view of, for instance, cT and alpha parameter intervals of sampling and number of sampled values. β and L show, for example, that there are four defects in the data set, since the points are four. (b) An example of M-TFM images, right: simulated ($\mathbf{X} \in \mathcal{X}$); left: experimental acquisition ($\mathbf{Y} \in \mathcal{Y}$). The two images correspond to the same parameter's vector $\mathbf{p}$.

the input in the trained DNN $u_{\boldsymbol{\theta}_*}$ to generate $\hat{\mathbf{Y}}_j \in \mathcal{S}$. Since $\mathbf{Y}_j$ is our GT, we can compute the error histogram over the test set for echo reconstruction metrics introduced at section III.2.2. Fig. III.7 represents the echo amplitude error frequency for the 100 samples of the test set. The echo position error is also calculated by the Euclidean distance in the (x, z) plane of the ROI. In Fig. III.8, a single instance reconstruction on the test set is shown in detail to illustrate the reconstruction error impact in a qualitative view.

A second evaluation over the $\mathcal{S}$ set is done to test the generative capabilities of the DNN. This time, a $\mathbf{X}_i \in \mathcal{T}$ and a $\mathbf{p}_j \in \mathcal{S}$ are picked up. The couples $(\mathbf{X}_i, \mathbf{p}_j)$ are forwarded from the input in the trained DNN $u_{\boldsymbol{\theta}_*}$ to generate $\hat{\mathbf{Y}}_j \in \mathcal{S}$. $\mathbf{Y}_j$ is our Ground Truth (GT) labelled by $\mathbf{p}_j \neq \mathbf{p}_i$. Then, we can compute the error histogram over the test set on echo reconstruction metrics (Fig. III.7). The results show the capability of the cU-Net as a surrogate model for new instance generation.

This test is a quantitative evaluation of data generated over a regression $\boldsymbol{p}$ input vector. Since only regular sampling was done over the parametric $\mathcal{P}$ to create the entire data set, the references of experimental M-TFM images for this evaluation are limited to the known $\mathbf{p}$ values. The selection of the

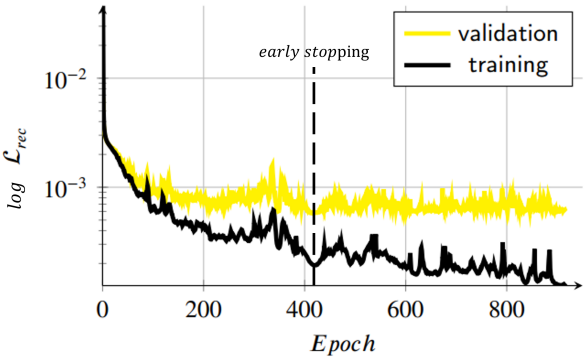


Figure III.6. Reconstruction loss evolution versus epochs for cU-Net training on M-TFM data set. The training loss is in black, and the validation loss is in yellow. The early stopping at 417 is marked by a vertical line. The training is stable even after the beginning of the over fitting. The reduced amount of epochs shows that few training samples can be used to train the architecture conditioned by parameters and at the same time get good reconstruction results, shown later in the test phase validation.

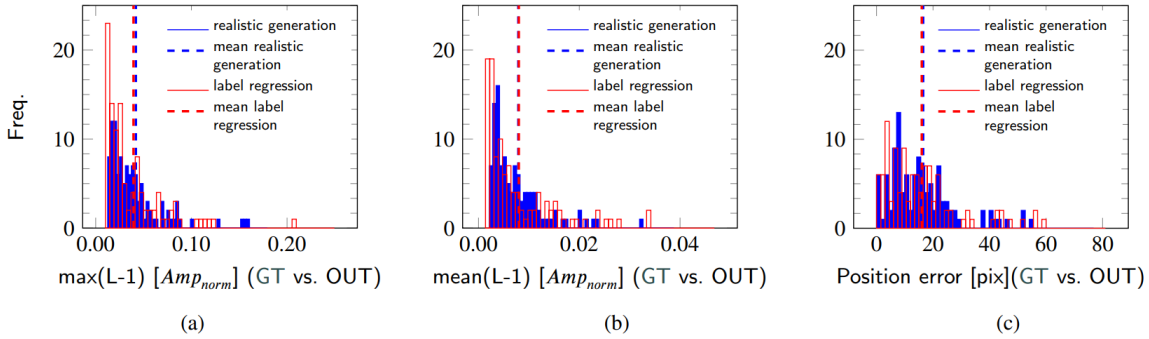


Figure III.7. Error frequency on the test set for realistic data set generation in blue. Generation error frequency on reconstruction parameters regression are plotted in red (described in section III.3.1). (a) shows the maximum of the pixel-wise difference between the output and the GT, (b) is the mean of the norm-L1 between the output and the GT, and (c) is the Euclidean distance in pixels between the maximum pixel of the output and the maximum pixel of the GT.

instances of $\boldsymbol{x}_i$ (TFM input simulations) for this evaluation are chosen by the following criteria: the label $\mathbf{p}_i$ of $\mathbf{X}_i$ is the closest to $\mathbf{p}_j$, in the $\mathcal{P}$ space. The metric used in $\mathcal{P}$ is the Euclidean distance between $\mathbf{p}_i$ and $\mathbf{p}_j$. This metric promotes the generation by reconstruction parameters regression (c_T and α). Thus, a regression over the flaw geometry parameters (β and L) was only performed to evaluate the generation. Both tests reported similar histograms (red bars in Fig. III.7). See the complementary video to illustrate conditional generation[2].

The described evaluation was designed to show the potential of the DNN for a surrogate model application. The model learns the link between the labels and the features required at the output, guided by the pST and FiLM layers. The following sections are focused on the internal features exploration of the DNN for the test set with the intention of describing how the parametric regression can be done in all directions of $\mathbf{p}$ to re-sample the $\mathcal{P}$ beyond the initial data set points.

III.3.2 . Analysis of the feature maps

[2] https://github.com/geragranados/M_TFM_cUNet

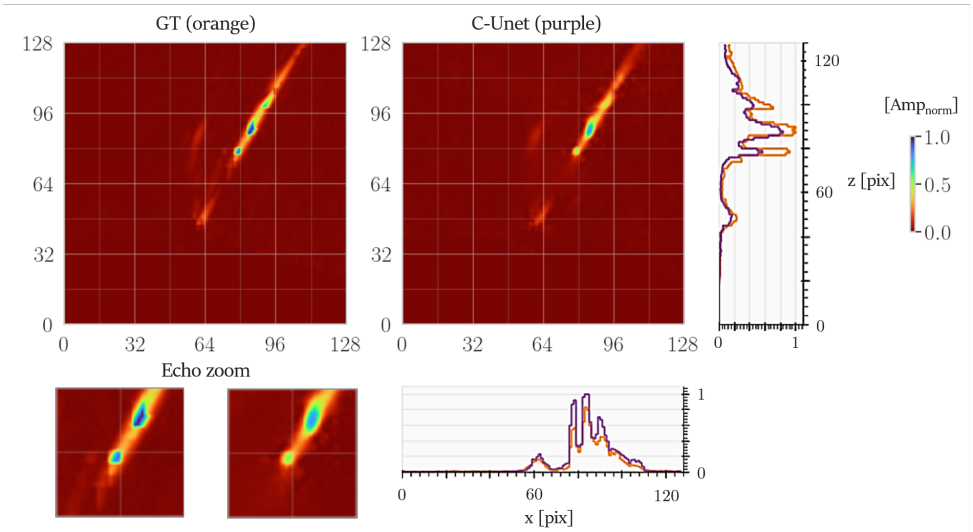


Figure III.8. Experimental generation example. Qualitative comparison between the output $\hat{\mathbf{Y}}$ and the GT, given a couple $(\boldsymbol{x}, \boldsymbol{p})$ at the input, where $\boldsymbol{p}$ is the correspondent set of parameters used to create $\boldsymbol{x}$ from the simulation. (a) is the maximum pixel for the line or column between the output and the GT. The maximum pixel amplitudes are plotted in purple for the output and in orange for the GT. Both images are overlapped for the ROI representation. In (b), the echo reconstruction shows the misfitting in terms of echo position and amplitude for the example given.

We observed how the conditional FiLM-pST layers act during the latent space generation when the scalar input p changes, an example of forward propagation of one image is shown in Fig. III.9. The output for each column (purple box image) can be contrasted with its GT (below the column) to evaluate the coherence of the conditional generated image. This evaluation is done over a sample of the test-set.

We can observe that the conditional layers make the filter implant new features in the activation maps; at the same time, they extract the main features of the input image. The intuition here is that the encoder is capable of creating new conditional features on the activation maps from the input image; these new features are conditioned by the input label values. Consequently, the image synthesis is also conditioned by the input label values. We can infer that the variety of possible experience-like images (outputs) that the decoder can create is linked to the range of the parameters observed during the training (see Fig. III.5).

Fig. III.9 also shows the behaviour of the DNN without the pST layer. This is done to show the role of each block: FiLM acts in the style content (e.g., SNR), and the pST acts in the context (echo shape). The last column shows that the FiLM ignores the echo translation linked to the new cT input value. At the same time, we can observe how a roto-translation is present between the first and second column due to the pST module from the second convolutional operation.

III.3.3 . Latent space exploration

A view into the output of the latent space block gives some clues about how the encoder works. When a M-TFM image is forwarded into the encoder, the features of the last layer of the latent space block represent a plausible high-dimensional representation of the input. This representation is obtained for each sample to generate the $\mathcal{Z}$ space. Since we are interested in the structure of this high dimensional space (i.e., 8x8x256 dimensions), we choose a t-SNE manifold projection [70] to reduce this space to a latent space (LS) representation in 2D. A principal component analysis initialisation is used for the t-SNE: the initial 2D coordinates are given by the two principal component values of the LS space, t-SNE is then initialized in a deterministic way.

Fig. III.10 represents a scatter plot of the $\mathcal{Z}$ space. A projection from $\mathcal{X}$ conditioned on $\mathcal{P}$ space

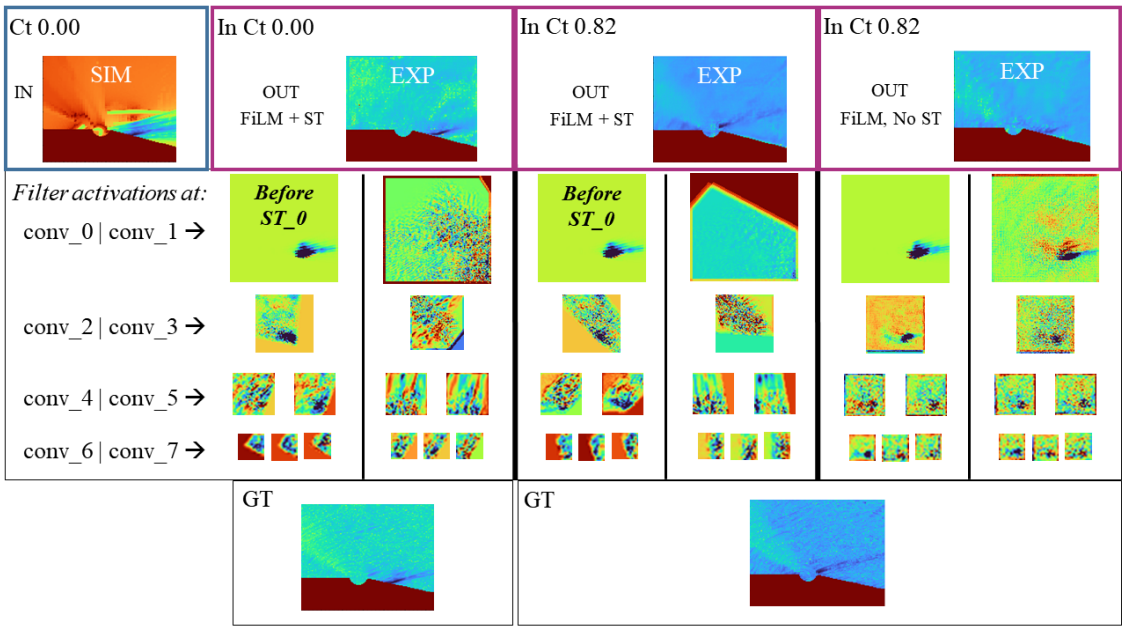


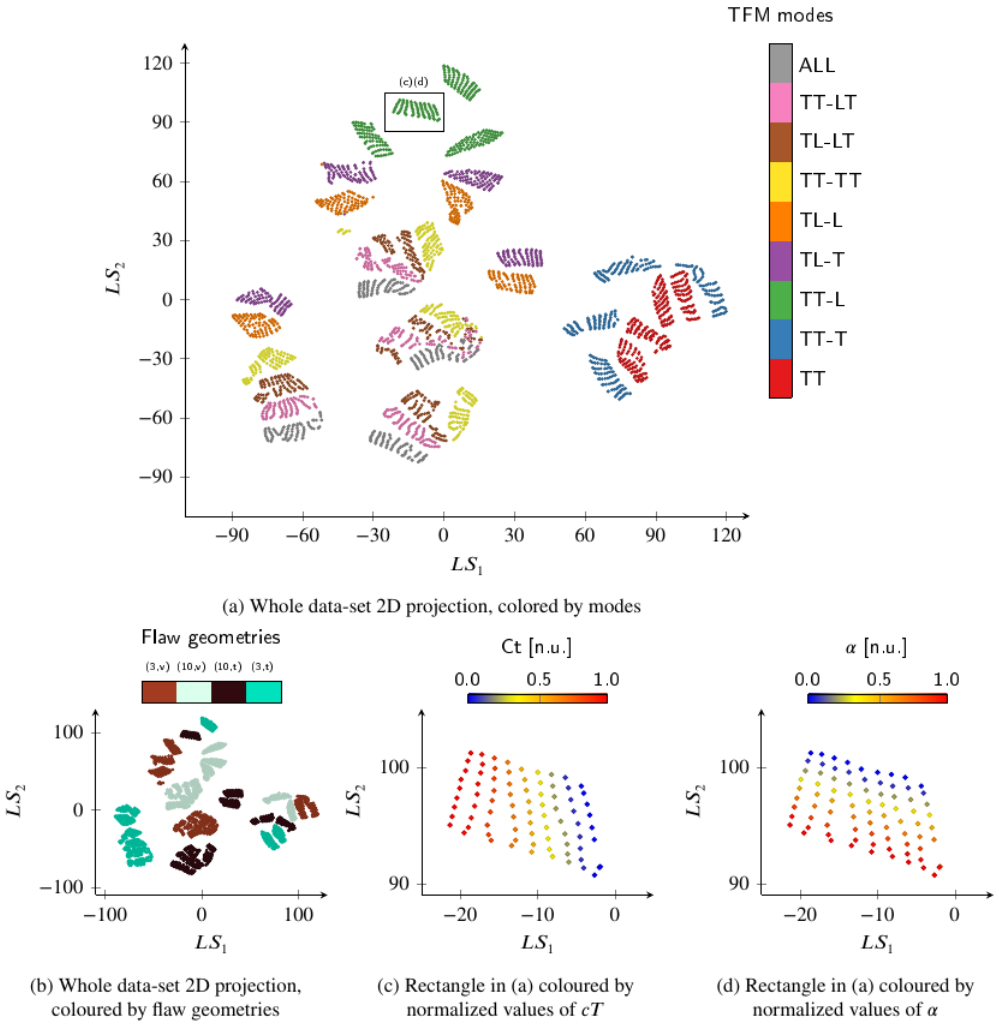
Figure III.9. The first column shows the activation values for the filters at different resolutions on F when the realistic generation is evaluated. At the end of the column, the GT image is shown for a qualitative observation. To better appreciate the changes in style, the input and output images are reported in logarithmic scale. The activation maps are normalised for better visualisation. For the second column, a conditional generation by changing the cT input value is evaluated. In this case, the same input as the first column is kept. For the third column, the same parameter was varied (cT), but this time by ablation of the pST blocks. The images from the layer activation are normalized by instance so the difference in pixel intensities can be observed. Each of the four activation map are transformed to a RGB format in the range of 0 to 255 (Min-Max).

to $\mathcal{Z}$ space is done by the encoder. Every z point is located on the 2D manifold by its coordinates (LS_1, LS_2). Consequently, each point can be related to a parameter value and input image. The representation in Fig.III.10 shows how the DNN builds $\mathcal{Z}$ in terms of clusters ordered accordingly to the different parameters.

That is, the first hierarchy observed in the manifold is the reconstruction modes. This order in the hierarchy is expected since the images in each reconstruction mode are significantly different. If we observe the parameters of the flaw's geometry, four main clusters are found in the manifold. When looking closer at one of these manifolds, an arrangement with respect to the two last parameters in $\mathcal{P}$ can be observed. Two mostly orthogonal directions are present for the celerity T and back-wall angle values.

The exposed structure of the $\mathcal{Z}$ space in this section, together with the exploration of the features, shed some light on how the DNN learns to generate data. The observations proved that the DNN creates a structured representation of the whole data set. The $\mathcal{Z}$ sampling in the right direction with respect to the parameters of the simulation is possible through the conditional encoder. This is how the input vector can generate other samples present in the data set, but also some samples not seen before. Those samples generated by the label regression are obtained by a consistent position in $\mathcal{Z}$. During the generation by regression, the encoder features are modified in the coherent direction, as is shown in the $\mathcal{Z}$ projection.

Given a fixed simulation image and a new label value never seen, the encoded 'point' in the latent space is then passed to the generator. The generators learnt to decode the known points of $\mathcal{Z}$, as it was shown in the section III.3.1. It is also expected for the generator to know 'how' to generate an intermediate point with similar characteristics to the neighbours but modified to be consistent with the new label value.



(a) Whole data-set 2D projection, colored by modes

(b) Whole data-set 2D projection, coloured by flaw geometries

(c) Rectangle in (a) coloured by normalized values of cT

(d) Rectangle in (a) coloured by normalized values of α

Figure III.10. c-Unet latent space projection into a 2D t-SNE manifold. The point represents the last layer activation of the green block in the c-UNet (Fig. III.3), just before the decoding. (a) shows the clustering by modes. (b) shows the clustering by flaw geometries. (c) and (d) is the a selected cluster (black rectangle in (a)) colored by two of the parameters: cT and α. t-SNE hyper-parameters: early exaggeration= 12, perplexity= 50. References for flaw geometries coloring: $(3, v)$ is the 3 mm tilted flaw geometry, $(10, v)$ is the 10mm vertical flaw geometry, and so on.

III.4 . Conclusions

In this work, we present a conditional UNet (cU-Net) for fast and realistic generation of multi-modal total focusing method (M-TFM) images. Our DNN is trained on both numerical and experimental data. As a result, our generative framework can learn realism from experiments along with a controlled generation learnt from the physics encoded in the simulations. We show how the cU-Net model performs the feature extraction to generate new realistic data. This is done by exploring the inner activation layers for different inputs. We also demonstrate how the Spatial Transformer Layer and Fidelity Layer Modulator provide the means to perform a regression in the M-TFM images by a cU-Net architecture.

The present framework allows the inclusion of simulation parameters directly as input in the surrogate model architecture. Once the model is trained, data generation can be done in quasi-real-time and guided by the input parameters regression. In doing so, the generated data are already labelled. However, the prior information on data fidelity levels (simulation and experience labelled couples) can be costly to

produce in some cases, and our approach turns out to be not highly efficient when this characteristic is not available in the data set. Nonetheless, the present architecture provides a way to fully inform a data generator DNN by the data labels.

Given this choice in the DNN architecture used for TFM image generation, we are not in the presence of a physics-agnostic NN. Moreover, the DNN is constrained to learn the physics behind the mapping between the simulated parameters and the TFM images. This is done by the pST and FiLM operators, as showed in the Fig. III.9 by the ablation of the pST during the generation of new images with different celerities. The reconstruction loss also plays a role for this task during the training, since the conditional layers are optimized to generate a consistent experience image given a set of parameters. To summarize, we embed the physics knowledge in the DNN architecture by means of the data simulated by the numerical solver, and the conditional generation driven by the parameters used in the simulation. Our approach is in contrast to more common data-driven approaches that are in the literature of ML as applied to image or signal processing communities.

An extensive analysis of the results by changing simulation parameters has been embedded in Appendix VIII.

Scalar labels were used as input in this work, so the inner sub-neural networks for the FiLM and pST Layer are dense layers. Those sub-neural networks are not limited to scalar inputs, but they can any shape. For instance, a CNN or RNN can be deployed to take into account other formats of labels (images, time series, etc.) that would guide later the generation.

III.5 . Chapter outlook and perspectives

Fast and reliable surrogate models have been used in NDT&E to speed-up the computational time for very intensive statistical studies ranging from the stochastic inversion [100] to the sensitivity analysis [112] and model-assisted probability of detection. In these application fields in NDT&E, our contribution aims to enhance the quality of surrogate model results making them more-close-to-reality. Therefore, by the use of the ML schema developed, the aforementioned studies will better account the impact of measure-like uncertainties in the studies outcomes.

This approach presents a first attempt at a physics-based and explainable surrogate model aiming at providing a controllable data generation tool. The new data can be exploited as a training set for deeper architectures (e.g., generative adversarial network) that demand big data sets, rarely available in the NDT&E field. Inversion problems may also benefit from an enlarged training set generated by this tool.

In our application case, we include the multi-fidelity data contribution present in the data set, but the framework can be adopted even if a single fidelity level is available in an auto-encoder architecture (no skip connections required). For instance, an experimental campaign may be enlarged by this DNN when it is correctly labelled.

In the perspectives of this work, we expect to explore the power of our cU-Net backbone architecture for inverse problems, given that the structure of the latent space seems promising for this task. Additionally, one notable mention is the possibility of using the cU-Net architecture to enhance the performance of simulation software widely used in modern NDT&E design problems. For instance, the low fidelity source of data may come from coarse meshed finite elements simulations or semi-analytical simulations, whereas the high fidelity from fine meshed finite elements simulations, among others.

IV - Development of a semi-supervised generative domain-adapted model in a TFM multi-fidelity dataset for enhanced automated inspections

About this section

In this chapter, the objective is the development of a *non-supervised* DL approach for TFM *multi-fidelity data set* generation without prior information on the data coupling or labels. A new approach that does not rely on this prior information is preferred for the objectives of this thesis, since many of the NDT&E data sets do not count with this characteristic (e.g., poorly labeled data). The same data set in chapter III is used to train a DL architecture, but this time by using less a priori information. The architecture learns to couple the fidelity by a probabilistic approach. The generation capabilities of the architecture are evaluated. An inverse problem is assessed with the realistic data obtained from the surrogate model to prove its efficacy.

IV.1 . Introduction

The pursuit of efficient methods to enhance the automation of NDT&E inspections driven by artificial intelligence is a topic of increasing interest in the community [9, 10].

To obtain a robust and accurate DL model for a NDT&E inspection, one needs a sufficiently large set of 'reliable' training data. Even if large simulated data set creation is nowadays possible and highly reliable simulations for NDT&E inspection are accessible, simulations barely reproduce the complexities of experimental or in situ data which may embed spurious contributions due to the environmental and experimental conditions.

To mitigate such a problem, a possible useful scenario is to employ a multi-fidelity data set when it is accessible. In this work, we refer to "fidelity" as the realism compared to the in situ data. In other words, how close data are to the real world data. Here, experimental data are the highest fidelity level available in the NDT&E procedure analyzed.

The natural approach when multiple fidelity levels are available is to try to use the more reliable data (highest fidelity level) as training data so the deployment of the automatic inspections reports a better performance. Unfortunately, sometimes the highest fidelity production details are not accessible or very scarce (confidentiality, poorly labeled, unknown data set production conditions).

Very recently, many methods have been developed to handle this problem in the field of NDT&E to take advantage of the full knowledge of this type of data set. Transfer Learning (TL) is a common approach when a similar task (e.g., image feature extraction) has previously been done in a completly different domain. This method takes advantage of previously trained NN to facilitate the training of a new NN [11, 12, 13] by reusing some layers. TL can be effective when data from a single fidelity level is present during the training to outcome better performance even when the data set is far from the in situ data in terms of distribution and the data set is small compared to the complexity of the task. A succeeding approach from TL is the Domain Adaptation (DA), suitable when more than one fidelity level is available. Nonetheless, it is often observed in NDT&E data that the closer to in situ data, the less information about how this data was produced is available, as discussed in [14, 15]. This lack of information translates to a poorly labeled high-fidelity data set. To circumvent such an issue, DA mixed with generative strategies or domain-adversarial strategies [16] have been tested in those cases to enhance DL algorithms for NDT&E inspections [18, 19, 20, 17]. DA may be applied to different situation that can vary from a large access of labeled data in many fidelity levels to just one fidelity level correctly labeled.

In our scenario, the source domain can have its origins in a simulated NDT&E procedure while the

target domain is the experimental data where an inversion ML algorithm is intended to be deployed. In other words, two or more degrees of high-fidelity data are accessible during training (e.g.: higher reliable simulation, experimental or in situ data).

In this case, a feature-based deep transfer learning can be applied to tend to reduce the difference between the two fidelity levels. In feature-based deep transfer learning, adversarial-based domain adaptation leverages differences in data distribution to acquire knowledge for the DA task [91, 15]. Specifically, this work is based in a generative adaptation model approach implemented with a Class-conditioned Generative Adversarial AutoEncoder (cGAAE) architecture.

In the present work, we propose a DL generative adversarial approach to learn the gap between two different fidelity levels when poor access to the data in terms labeling or quantity. The proposed strategy falls within the DA technique, relying on a realistic data generator based on a deep neural network (DNN). The final objective is to generate an enlarged realistic data set when high-fidelity data labels are not available and enhance the performance on a targeted task. Toward this end, we train a tailored NN architecture aiming at maximizing both reconstruction and adversarial losses. More specifically, the ensemble of DNNs constitute what we call a cGAAE, based on the architectures from [149, 150], among others, where discriminators are included to guide the realistic generation and the cGAAE latent space structure. In this scenario, the cGAAE takes a source sample and a latent vector, called $\boldsymbol{z}$-vector, to generate a new target domain instance. The vector $\boldsymbol{z}$ is a domain-adapted representation for the source and target domains. This representation is learned by an encoder and assisted by a domain classifier. The cGAAE also uses a reconstruction-based training loss to ensure consistency generation. This hybrid feature-base method is intended to translate a source domain to a closer representation of the target domain.

The generator uses as input the fidelity level (identified by a discrete category or class in this work) as conditioning. Once trained, during the on-line stage, the choice of the condition input of cGAAE is used to generate realistic data from a low-fidelity data (or vice-versa). Furthermore, we introduced a class-conditioned Adaptive Instance Normalization (AdaIN) based on the instance norm layer in [82] and a Class-conditioned Spatial Transformer (cST) as a variant of conditional ST from Chapter II and Chapter III. Those modules are used in the architecture to ensure the class-conditioning of the cGAAE. They also ensure the generation task in a frame of a small data set by allowing a smaller and simplest architecture in terms of training parameters, comparable to the similar architectures in the bibliography.

For the proposes of this work, the framework is deployed with a two-fidelity data set containing Multi-modal Total Focusing Method (M-TFM) images from the UT inspection of a complex geometry welding joint. The generated data are validated in an inversion problem DL benchmark to asses its performance when the training data are more realistic. In this work, we focus in a high definition image generation such it can be observed in [151, 152, 82] to bridge the gap between the state-of-the-art in DL generative networks and data production in the field of NDT&E. It is worth to be mentioned that the proposed framework has been applied in the computer vision research community to ensure the quality generation such in [151, 118, 82].

This Chapter is structured as follows: Section III.2.1 describes in detail the data production used for the architecture training and testing. Section III.2 describes the methodology, the proposed architecture and the algorithm to train the architecture. Section IV.4 presents some results in generation and a benchmark performance assessment when the generated data are used in an inverse problem. Section IV.5 discusses the architecture choices, stability issues and solutions found to successfully train the generative neural network framework proposed. Section IV.6 presents the conclusion and perspectives of this work.

IV.2 . Semi-supervised generative adaptation model in a TFM multi-fidelity dataset

IV.2.1 . TFM multi-fidelity dataset and enlarged simulation

The data set is composed by Full Matrix Capture (FMC) simulated and experimental acquisition (a detailed description of the data set used in Chapter III). The resulting acquired temporal signals are post-

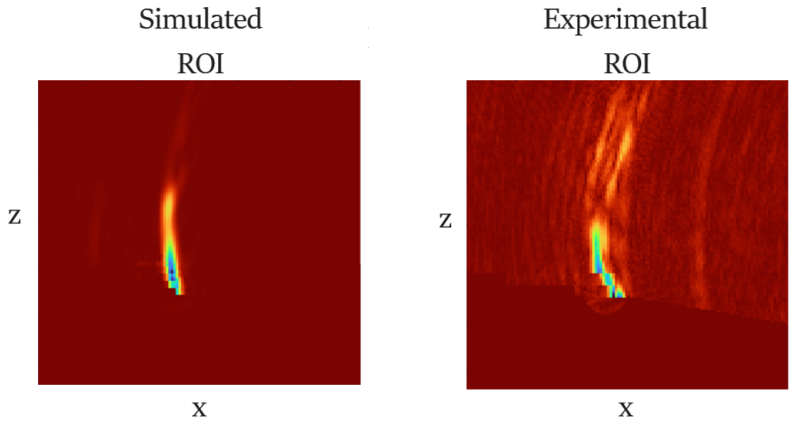


Figure IV.1. The M-TFM data set contains a set of images labeled by the simulation parameters and classes. The figure show an example of M-TFM images, simulated and from acquisition. The two images correspond to the same parameter's vector $\mathbf{p}$ and different class $\mathbf{c}$ (i.e., simulated vs. experimental images on the left and the right, respectively). One can observe the impact of the non-quantified uncertainties between fidelity levels.

processed by the M-TFM imaging algorithm [153] in a defined Region of Interest (ROI). The mentioned acquisitions produce a set of images of high resolution with shape $[M, W, H, C_{modes}]$, where M is the number of acquisitions, W, H, C_{modes} are the pixel weight, pixel height and number of channels of the images, respectively. C_{modes} here represent different reconstruction modes as described in [153] for a same FMC acquisition. It is worth noting that the multiple modes used can be seen as the equivalent of a hyper-spectral images in computer vision and remote sensing communities.

The imaging data set contains two fidelity (interpreted as fictitious classes in this work), the physics-based M-TFM simulation images as lower fidelity level and the experimental images as higher fidelity level. In terms of DA, we account for two source domains of a different fidelity level each one, where the lower fidelity is the source domain and the experience is the target domain.

Since the data set mixes both fidelity levels, a fidelity level class label $\mathbf{c}_n$ (two classes) is added to each sample together with previous labels to distinguish each N-samples fidelity level. That is, the final data set can be expressed as $[N, W, H, C_{modes}, C]$ with $N = M \times C_{modes} \times 2$, when each of the modes is considered separately with the two classes associated to each of the modes. The resulting single channel and multi-class data set $\mathcal{D}$ as given in Eq.IV.1, is used to train the described generative architecture in the next sections.

$$\begin{aligned} \mathcal{D} &= \{(\mathbf{X}_n, \mathbf{c}_n) : n \in \{1, ..., N\}\}, \\ \mathbf{X}_n &\in \mathcal{X} \subseteq \mathbb{R}^{W \times H \times C},\ \mathbf{c}_n \in \mathcal{C} \subseteq \mathbb{Z}^C. \end{aligned} \tag{IV.1}$$

It is worth noticing that $\mathbf{X}$ represents a TFM image of both fidelity levels. Contrary to the Chapter III, here the multi-fidelity data are mixed into a single set $\mathbf{X}$ sorted by the class label $\mathbf{c}$. $\mathbf{c}$ informs the fidelity level in the data set. This nomenclature is more adapted for the generative model nomenclature.

During the FMC post-processing applied in both fidelity levels, some variation in the parameters of M-TFM reconstructions where applied. The celerity of the transverse ultrasonic wave c_T and the specimen back-wall slope angle α (Fig. IV.2) are considered to lead to major impacts into the M-TFM reconstruction. One can account such an impact on the experimental data by applying the M-TFM algorithm repeatedly to FMC acquisitions with different reconstruction values of c_T and α. It is noteworthy that such a way of generating samples based on FMC acquisitions can be seen as a data-augmentation

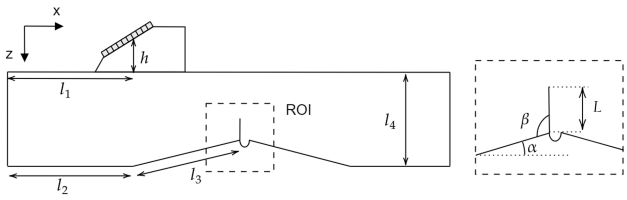


Figure IV.2. Sketch of the mock-up configuration with the main parameters used to describe it.

based on physics-rooted principles. The sampling values of these parameters are kept as labels for each image instance for later proposes.

The other parameters take part in the labels of the images, the flaw geometry present in the mock-up is also simulated and they are expressed by the height (L) and the tilt (β). Those parameters are named as $\mathbf{p}_n \in \mathcal{P} \subseteq \mathbb{R}^{n_p}$, where n_p is the number of parameters. They are not used in the training of the generative framework, but used later for the analysis of the trained cGAAE and the aimed inversion task. As already mentioned in the introduction, the experimental acquisitions are impacted by uncertainties that cannot be observed directly while measurements are performed. Indeed, they can have origin in the experimental conditions (e.g., environmental conditions), probe position, human factors, acquisition noise (e.g., electronic noise), among others. This kind of uncertainties may have a non-negligible impact in the M-TFM image produced in face of a simulated one (Fig. IV.1). The proposed DA procedure relying on cGAAE has as its main purpose to fill the gap of discrepancy between simulated and experimental data by generating images embedding realistic high-fidelity patterns.

IV.3 . Insight on the Class Generative Adversarial AutoEncoder Network Architecture

The present work proposes to train a cGAAE architecture with a high-dimensional multi-fidelity source data set $\mathcal{X}$. The filter (or encoder) $F : \mathcal{X} \rightarrow \mathcal{Z}$ encodes $\mathbf{X}_n$ into $\mathbf{z}_\mathrm{n}$, so $\mathcal{X}$ is mapped into a latent space $\mathcal{Z} \subseteq \mathbb{R}^d$ by F, being $d << W \cdot H \cdot C$. $\mathcal{Z}$ is reduced domain-adapted representation of $\mathcal{X}$ where $F(\mathbf{X}_n) = \mathbf{z}_n$. The generator (or decoder) $G : \mathcal{Z} \times \mathcal{C} \times \mathcal{N} \rightarrow \mathcal{X}$ maps the latent representation $\mathcal{Z}$ to data space $\mathcal{X}$ using the class $\mathbf{c}$ as input (represented by a vector), a noisy input $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\epsilon})$ being $\mathbf{n}, \boldsymbol{\epsilon} \in \mathcal{N} \subseteq \mathbb{R}^d$, with $\boldsymbol{\epsilon} << 1.0$, and $\mathbf{0}$ is the zero matrix of dimension $(d \times d)$. F and G generate $\hat{\mathbf{z}} \in \mathcal{Z}$ and $\hat{\mathbf{X}} \in \mathcal{X}$, respectively, where hat notation $\hat{}$ identifies the generated sample either from F or G. This approach is similar to the Generative Adversarial Network (GAN) architecture found in [154] and the generator is an adapted architecture from [151].

While F and G are trained, a set of discriminators are optimized in an adversarial way to compute the scores $s_x, s_{xz}, s_z \in \mathbb{R}$. As instance, the image discriminator $D_x : \mathbf{X} \mapsto \{s_x|\mathbf{c}\}$ discriminates a real image $\mathbf{X}_n$ from fake generated $\hat{\mathbf{X}}$ giving and score s_x as output. D_x output is used in an adversarial way so G leads generated images that look like the images on $\mathcal{X}$ for both classes in $\mathcal{C}$. To do that, D_x uses $\mathbf{c}_n$ and $\mathbf{X}_n$ to learn the distribution of $\mathcal{X}$ while judging if $\hat{\mathbf{X}}$ for a given class belongs to $\mathcal{X}$.

To get $\mathcal{Z}$ from $\mathcal{X}$, F uses an adversarial schema together with a latent space discriminator $D_z : \mathbf{z} \mapsto \{s_z\}$. Since $\mathcal{Z}$ is expected to be a sampling space to get $\mathcal{X}$ through G, a distribution is imposed to the latent space to obtain $\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, where $\boldsymbol{I}$ the identity matrix of dimension $(d \times d)$. The task of D_z is to discriminate real $\mathbf{z}_n$ from fake generated $\hat{\mathbf{z}}$ samples from F. An additional discriminator $D_{xz} : \mathbf{X} \mapsto \{s_{xz}|\mathbf{z}\}$ is used to distinguish between the two couples $(\mathbf{X}_\mathrm{n}, \hat{\mathbf{z}})$ and $(\hat{\mathbf{X}}, \mathbf{z}_\mathrm{n})$. D_{xz} is inspired from [150, 155, 151], where they assure a bidirectional mapping and a semantic meaningful latent representation learned in an unsupervised way. The adversarial losses described the Section IV.3.3 use the discriminators outputs as a score for the hinge loss [76] (defined later in Section IV.3.3) to classify the

generators outputs and the data set samples by creating a separating hyper-plane, similarly as Support Vector Machine (SVM) loss [156]. The discriminator are expected to converge to a Nash equilibrium [157], where the generators outputs are indistinguishable from the data set samples; in others words, the scores s_x, s_{xz}, and s_z are on average the same when the data are sampled from F and G, or from $\mathcal{D}$ and $\mathcal{N}$.

For simplicity, $\hat{\mathbf{z}}$ and $\mathbf{z}_n$ will be recalled in the next sections simply as $\mathbf{z}$, unless a distinction between generated and original samples is required. Same for $\mathbf{X}_n$ and $\hat{\mathbf{X}}$ who are notated as $\mathbf{X}$.

IV.3.1 . Deterministic filter and stochastic conditional generator architecture

In our architecture, the filter F is a deterministic function of $\mathbf{X}$. The objective of F is to learn an invariant domain latent representation $\mathcal{Z}$ for $\mathcal{X}$. The encoder output is a $\mathbf{z}$ latent vector for each $\mathbf{X}$. The latent space inner structure is guided by the adversarial game where D_{xz} and D_z impose a distribution of $\mathcal{Z}$. F a DNN with convolution ResNet-like skip-connections variant taken from [151] (Fig. IV.3) with average pooling layers to decrease the resolution of the features. The last layer is a dense layer that takes the flattened extracted features from the convolutions to produce a latent vector. The filter acts as a bottleneck to go from a high-dimension feature representation of size $D = W \cdot H \cdot C$ to a low-dimension feature representation d. Since F has no additional stochastic input, it turns to be a deterministic function that accommodates the extracted information from the existing samples in $\mathcal{X}$ in a $\mathcal{Z}$ reduced space in a parsimonious embedding representation. In other words, all the information from the M-TFM images is compressed in a domain-adapted representation from which the generator synthesizes a new image.

The generator G architecture also implements the ResNet connections as the filter, but with up-sampling layers instead of average pooling (Fig. IV.4) to progressively increase the resolution. $\mathbf{z}$-skip connections (shown in green arrows in Fig. IV.4) are used when forwarding $\mathbf{z}$ through G, similar to [151, 82]. As a result, $\mathbf{z}$ latent vector is split in s partitions $\mathbf{z}_i$, where s is the number of ResNet blocks in G and $i = 0, ..., s-1$. $\mathbf{z}_0$ is forwarded through a dense layer before synthesizing the 2D features for the first ResNet block input. $\mathbf{z}_{1,..,s-1}$ are propagated differently. They are the input of conditional layers explained later. The generator uses convolution layers that output a tensor of activation maps or features. Those features are passed through Weighted Noise (WN) layers [82]. Class-conditioned AdaIN layers are placed before each convolution layer. All blocks have a ReLU activation function after each class-conditioned AdaIN layer. Additionally, the proposed cST is sandwiched between two ResNet blocks only once in G's architecture. To end up, the generator used a set of convolution layers with no skip-connections stacked with Instance Normalization (IN) layers and hyperbolic tangent activation functions and a WN at the beginning of the sequence. This part of the generator (purple blocks at Fig. IV.4(a)) is call the synthesis block adopting non-conditioned layers. This block provided a better reconstruction on the details of $\mathbf{X}$ when it was added.

The generator has one Self-Attention (SA) layer between two ResNet blocks such as in [158] at resolution $(W/4, W/4, 32)$. The localization of this layer, as the cST layer, is strategically set in a low-resolution stage in G, given that the number of trainable parameters and forward computation of these layers explode when the activation maps resolution increases. The details of the particular layers such WN, class-conditioned AdaIN and cST layer in G are commented later in this Section. The SA layer was added to G after empirical testing in G, driving to a better reconstruction for $\mathbf{X}$.

Here above, we have provided the details of the architecture on the different blocks constituting G based on the use of $\mathbf{z}$, $\mathbf{n}$ and $\mathbf{c}$ input vectors. In particular, $\mathbf{z}$ vector is intended to contain all the necessary information to reconstruct $\mathbf{X}$, while $\mathbf{n}$ is a noisy input whose stochastic behavior is kept during the training and test phase. The vector $\mathbf{c}$, is a static embedding vector for the present classes in $\mathcal{X}$ (e.g., an one-hot encoding). In our scenario, we only have two classes linked to two domains or data fidelity. $\mathbf{c}$ is used during the training phase of the cGAAE in a semi-supervised way. The corresponding class for $\mathbf{X}$ is forwarded into the generator when the reconstruction is intended. Moreover, $\mathbf{c}$ is switched to the opposite class and the resulting output $\mathbf{X}_{cs}$ is discriminated by the class projection D_x, so G learns to reproduce an opposite class for $\mathbf{X}$. The specific losses used for this task are explained in Section IV.3.3.

In the case of $\mathbf{n}$, this input is used as a source of noise applied to $\mathbf{z}$ during the synthesis of $\mathbf{X}$ and is

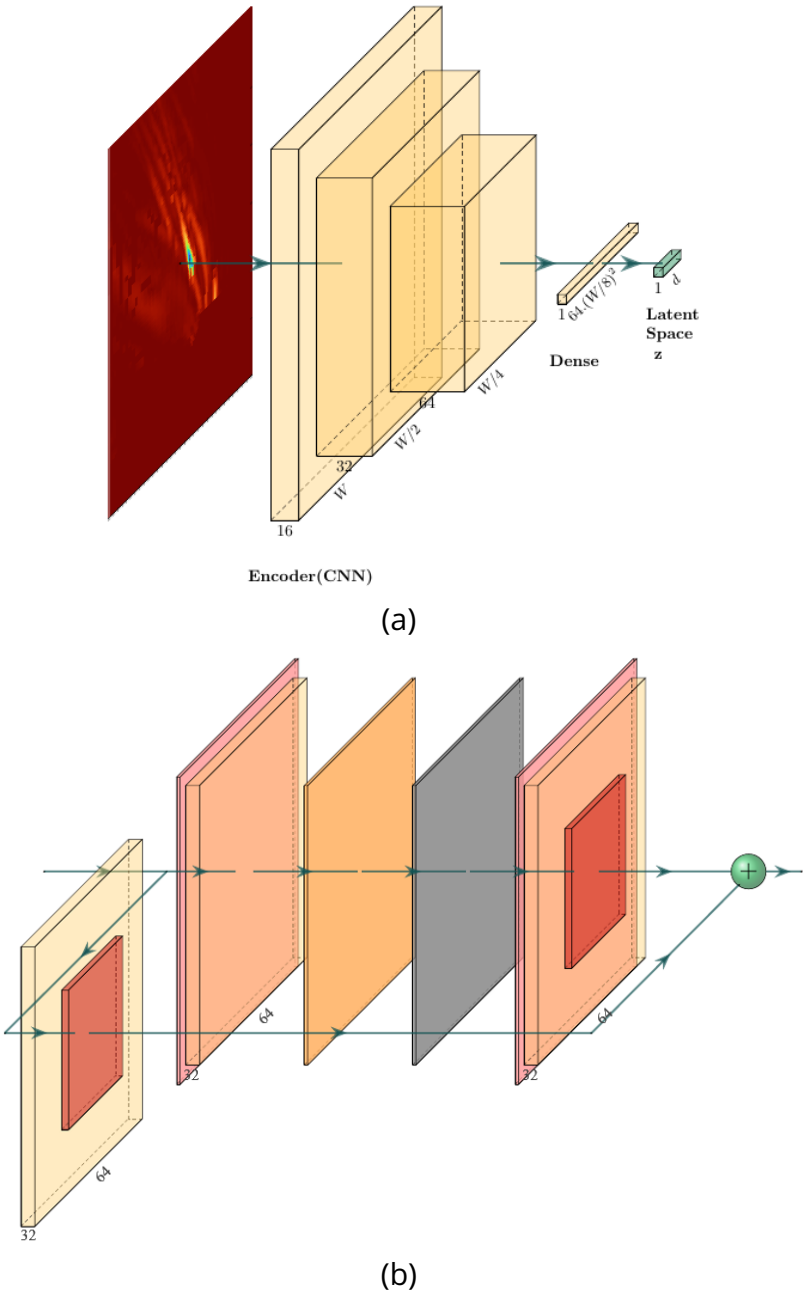
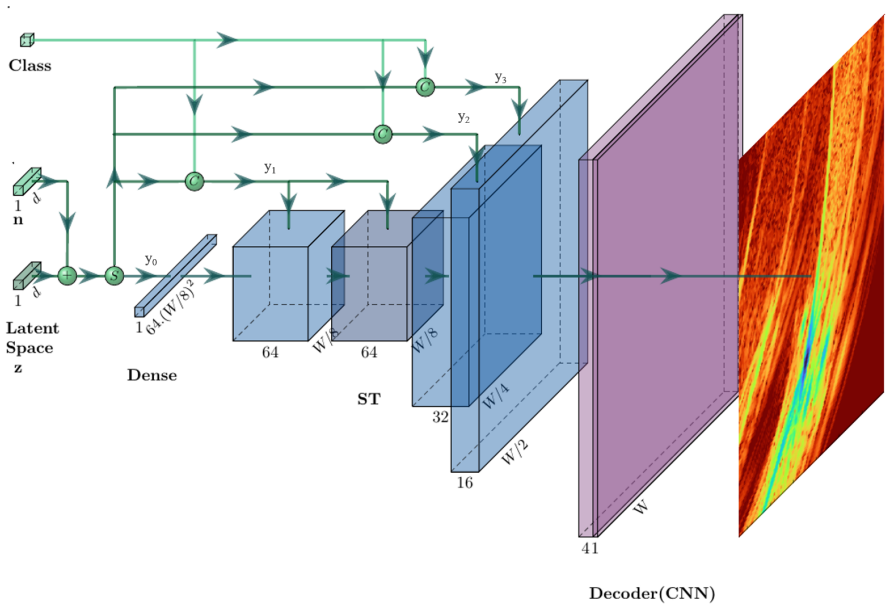
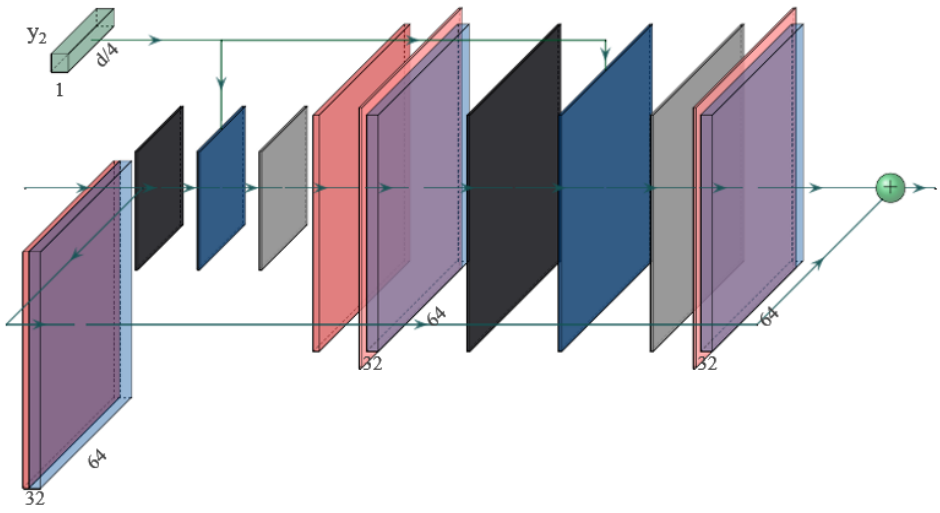


Figure IV.3. (a) Filter bottle-neck architecture, the first three yellow boxes represent a ResNet block at a different resolution stage in the feed-forward network. The thin yellow box represents the flatten operation to feed a dense layer. The output after the dense layer is a reduced latent representation of size d. (b) Filter ResNet block detail of the first three yellow boxes in (a), this is specifically the second ResNet block in the encoder at resolution of 64 pixels. The structure of a ResNet block contains the following layers from left to right at the main branch: a zero padding (**pink**), a convolution (**yellow**), an instance norm (**orange**), a ReLU activation (**gray**), a zero padding, a convolution, and an average pooling (**red**). The skip res-net connection has a convolution and an average pooling. The outputs of both branches are added at the end. Images generated with [146].



(a)



(b)

Figure IV.4. (a) Generator synthesis architecture, the 3 **blue** blocks (64,32,16) represent the ResNet blocks at a different resolution stage in the feed-forward network. In between the cST layer in **dark blue**. The first thin **blue** box represents a dense layer output and reshape operation to feed the first convolution layer. Synthesis layers in **purple** as the last layers for the generator with the number of filters equal to 4 and 1. (b) Generator ResNet block detail of the 3 blue boxes with 64,32,16 filters in (a), this is specifically the second ResNet block in the encoder at the resolution of 64 pixels. The structure of a ResNet block contains the following layers from left to right at the main branch: a Weighted Noise (**black**), a conditional AdaIN (**dark blue**), a ReLU activation (**gray**), an up-sampling (**red**), and zero padding (**pink**), and a convolution (**blue**). This arrangement is repeated once without the up-sampling for the main branch. The skip res-net connection has an up-sampling and a convolution. The outputs of both branches are added at the end. Images generated with [146]. *s* circle stands for the split operation and *c* circles for concatenation operation.

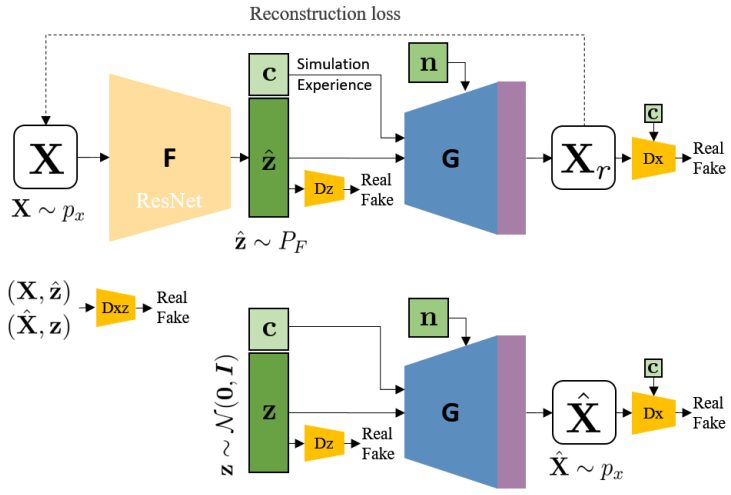


Figure IV.5. Adversarial auto-encoder scheme. Above, the chain of generation of reconstructed images $\mathbf{X}_r$ starting from a real image $\mathbf{X}$ is represented. The generation has an intermediate generation stage, getting $\hat{\mathbf{z}}$ from $F(\mathbf{X})$ before generating $\mathbf{X}_r$ from $G(\hat{\mathbf{z}})$. Below, the generation of fake$\hat{\mathbf{X}}$ coming from real z.

intended to produce stochastic reconstructions of $\mathbf{X}$, such in [159]. The key idea of adding $\mathbf{n}$ as input is to capture the variability in the data observed by the generator, particularly when a class change is intended. The sampling on $\mathbf{n}$ gives G a stochastic aspect to map one $\mathbf{z}$ vector to many $\mathbf{X}_{cs}$ images like in the implicit auto-encoders [160, 159].

Both generators, together with the discriminators, are used during training in an adversarial framework as shown in Fig. IV.5. The discriminator details are exposed in Subsection IV.3.2. Fig. IV.5 shows how "real" and "fake" examples of the latent space and the feature space (images) are feed into the discriminators. The flows of generation show how the different inputs of the losses (details in Section IV.3.3) are obtained during training by feeding different conditioned inputs into the generators and discriminators.

IV.3.1.1 . Class-conditioned spatial transformer and instance normalization

The concept of content and style in relation to generating or manipulating data, particularly in the context of images, is attributed to [161]. Under this assumption, an image contains two different sources of information: the content is the more important visual features (e.g. the eyes in a cat picture or a house in a landscape); in the other hand, the style is often related to the background, the luminosity, the contrast, the dominant colors and particularly, the high-frequency “noise” in the image. [161] demonstrate how the two parts, style and content, can be separated for generative proposes. In the case of [82], they show how the content and the style can be separated the latent representation depending on the stage (resolution) of the image synthesized at the generator.

In this work, the M-TFM image generation is also focused on the separation of content and style. By observing the Fig. IV.1, it is possible to observe differences in the content (e.g., echo in the image) and in the style (e.g., the electronic noise in the background or the Signal-to Noise Ratio (SNR). Since this approach focuses on a high-definition image generation on both fidelity levels, it is worthy that G is capable of reproducing both content and style. This is relevant since a gap between fidelity levels in the data set can be observed in terms of content and style. The class-conditioned generator must be able to synthesize both characteristics in the image depending on the class.

One of the proposed solutions for this task, especially for the content translation, is the cST layer. This layer acts as the content translator between classes during the training and the testing phase of the cGAAE. Spatial transformers were introduced by [117] to enhance CNN classification task by learning how to eliminate spatial shift, rotations, scaling or shearing in the input data. Here, the localization

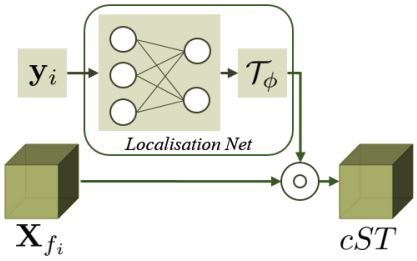


Figure IV.6. Class-conditioned spatial transformer with a dense localization network.

network is modified as shown in Fig. IV.6 to take as input a partition $\mathbf{z}_i$ of $\mathbf{z}$ and the class embedding $\mathbf{c}$ (i value depends on the position of the cST layer G). The objective of adding this layer is to facilitate the task of content changing by the $\mathbf{c}$ and content synthesis by $\mathbf{z}_i$. Furthermore, this layer input is affected by the latent noise $\mathbf{n}$ since it is added to $\mathbf{z}_i$ before going trough the localization network in the cST. As a result, $\mathbf{n}$ is expected to capture the variability of the images linked to the content. We denote $\tilde{\mathbf{z}}_i$ is the result of $\tilde{\mathbf{z}}_i = \mathbf{z}_i + \mathbf{n}_i$, in order to sample different instances from the same point at $\mathbf{z}_i$ in the latent space by sampling $\mathbf{n}_i$, promoting diversity in generated outputs. Additionally, adding $\mathbf{n}_i$ encourage the model to learn a more robust and smooth representation.

For any n-th sample in $\mathcal{X}$, the cST layer, placed after the i-th ResNet block, uses a dense layer as a localization network to infer $\boldsymbol{\phi} : \mathbb{R}^{6\times C_i}$ from $\mathbf{y}_{i_n}$ so $T_{\boldsymbol{\phi}} : \mathbb{R}^{3\times 3\times C_i}$ can be built, where C_i are the channel number of the input feature map $\mathbf{X}_{\mathrm{f_i}n} \in \mathbb{R}^{\mathrm{W_i}\times \mathrm{H_i}\times \mathrm{C_i}}$ and $\mathbf{y}_{i_n}$ is the concatenation of $\tilde{\mathbf{z}}_{i_n}$ and $\mathbf{c}_n$, so $\mathbf{y}_{i_n} = [\tilde{\mathbf{z}}_{i_n}; \mathbf{c}_n]$. $\boldsymbol{\phi}$ and $T_{\boldsymbol{\phi}}$ for each C_k are built as

$$\begin{gathered}\boldsymbol{\phi}_{C_i}(\mathbf{y}_{i_n}) = [\phi_{1_{C_i}}(\mathbf{y}_{i_n}),\ \phi_{2_{C_i}}(\mathbf{y}_{i_n}),\ \phi_{3_{C_i}}(\mathbf{y}_{i_n}),\\ \phi_{4_{C_i}}(\mathbf{y}_{i_n}),\ \phi_{5_{C_i}}(\mathbf{y}_{i_n}),\ \phi_{6_{C_i}}(\mathbf{y}_{i_n})],\\ T_{\boldsymbol{\phi}_{C_k}}(\boldsymbol{\phi}_{C_k}) = \begin{bmatrix} \phi_{1_{C_i}} & \phi_{2_{C_i}} & \phi_{3_{C_i}} \\ \phi_{4_{C_i}} & \phi_{5C_{C_i}} & \phi_{6_{C_i}} \\ 0 & 0 & 1 \end{bmatrix},\end{gathered} \tag{IV.2}$$

to be applied as an affine transformation per channel C_i to the feature map coordinates of $\mathbf{X}_{\mathbf{f_i}}$ as follows,

$$coord(\mathbf{X}_{\mathrm{f-st_i}n}) = T_{\boldsymbol{\phi}}(\boldsymbol{\phi}(\mathbf{X}_n)) \odot coord(\mathbf{X}_{\mathrm{f_i}n})\ . \tag{IV.3}$$

The resulting coordinates from Eq. (IV.3) are used to compute a transformed feature map $\mathbf{X}_{\mathrm{f-st_i}n}$. To do that, the pixel intensities in $\mathbf{X}_{\mathrm{f}n}$ are used together with the source coordinate grid $\mathrm{coord}(\mathbf{X}_{\mathrm{f}n})$ to get the pixel values for the new target grid $coord(\mathbf{X}_{\mathrm{f-st_i}n})$. A bi-linear interpolation is applied with the four closed neighbors from the source grid. The operation denoted by $\odot$ performs a matrix multiplication each set of (x, y) coordinates at $\mathbf{X}_{\mathrm{f_i}n}$ for each channel. As a result, $\mathbf{X}_{\mathrm{f-st_i}n}$ is a representation of $\mathbf{X}_{\mathrm{f_i}n}$ after the learned an affine transformation is applied at each channel. The resulting feature map from the cST layer is the input of the $(i+1)$-th ResNet block in G. The whole cST operation (Eq. (IV.2) and IV.3) is represented by $\mathcal{T}$ from now-on, witch is applied to $\mathbf{X}_{\mathrm{f_i}n}$ as follows,

$$\mathrm{cST}(\mathbf{y}_{i_n}, \mathbf{X}_{\mathrm{f_i}n}) = \mathcal{T}_\phi(\mathbf{y}) \circ \mathbf{X}_{\mathrm{f_i}n}. \tag{IV.4}$$

Regarding to the style, the class-conditioned AdaIN layer is proposed for the style translation task. This layer implements an IN operation who has been successfully implemented in several works for fast style transfer in GAN architectures [118, 162]. This layer is modified in our architecture to be informed not only by z_i like in [82], but also by the encoding of $\mathbf{c}$. The background idea of this choice is similar to the conditioning in the cST: the different classes have different styles. In other works, the noise and content distribution are different when looking to each fidelity side of the data. This layer used also a

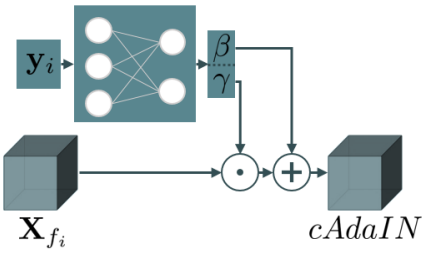


Figure IV.7. Adaptive instance normalization layer with a dense network for class conditioning.

$\mathbf{z}_i$ as input, so G takes into account the conditional F output $\mathbf{z} = F(\mathbf{X})$ either to reconstruct $\mathbf{X}$ or to keep the relevant information of $\mathbf{X}$ to produce the class conditioned $\mathbf{X}_{sc}$. Each class-conditioned AdaIN placed at the i-th ResNet block uses a dense layer to infer $\beta \in \mathbb{R}^{C_k}$ and $\gamma \in \mathbb{R}^{C_k}$ from $\mathbf{y}_{i_n}$ to apply, then

$$\mathrm{cAdaIN}(\mathbf{y}_{i_n}, \mathbf{X}_{\mathrm{f_i}n}) = \gamma(\mathbf{y}_{i_n}) \cdot \frac{\mathbf{X}_{\mathrm{f_i}n} - \mathbb{E}[\mathbf{X}_{\mathrm{f_i}n}]}{\sqrt{\sigma^2[\mathbf{X}_{\mathrm{f_i}n}] + \epsilon_{IN}}} + \beta(\mathbf{y}_{i_n}), \tag{IV.5}$$

where $\mathbb{E}[\cdot]$ and $\sigma^2[\cdot]$ represent the empirical average and variance, respectively. ϵ_{IN} is a small value used for numerical stability.

Both class-conditioned layers presented in this section give the degree of freedom to G to separate z-partitions by different hierarchical content and style. Similarly, $\mathbf{c}$ is fed to the layers independently, so each ResNet block can be pilot separately, either from $\mathbf{z}_i$ or $\mathbf{c}$. This was done with the aim of studying the behavior of the layers at each resolution level of G. The noise introduced by $\mathbf{n}$ gives G a stochastic nature for the generation, acting directly to the conditioning of both layer an so, to the content and style variation.

IV.3.1.2 . Weighted noise layer

A key part of the style synthesis for G is the Weight Noise (WN) layer that was originally presented in [82]. The authors show how this layer boost the random texture generation for image synthesis. Here, this layer was successfully implemented to allow G to reproduce the background noise in the images. The layer adds 2D Gaussian noise map randomly sampled at each forward step, during the training and the test phase. The noise is added to each feature map channel at each resolution level of G, after affecting the noise by a weight per-channel. The weights $\boldsymbol{B} \in \mathbb{R}^{1\times1\times C_i}$ are learn-able giving G the freedom to choose where the noise is added. This stochastic layer is intended to act only in the high-frequency noise reproduction in an aleatoric way. In future sections, we show how this layer plays in the background noise (or texture) generation.

It is worth mentioning that, in most of the cases, the texture in images from the NDT&E field can be interpreted as a stochastic process. Nevertheless, a variation of this layer configuration can be obtained by sampling just once the 2D Gaussian noise during the training so the synthesis of noise becomes deterministic, but not implemented in this work. For an input feature $\mathbf{X}_{\mathrm{f_i}n}$, the WN layer adds a noise map S, weighted by the B as follows,

$$\mathrm{WN}(\mathbf{X}_{\mathrm{f_i}n}) = \mathbf{X}_{\mathrm{f_i}n} + \boldsymbol{S} * \boldsymbol{B}, \tag{IV.6}$$

where $\boldsymbol{S} \sim \mathcal{N}(\mathbf{0}_{WH}, \mathbf{I}_{WH})$ and $\mathbf{I}_{WH}, \mathbf{0}_{WH} \in \mathbb{R}^{C_i}$. The operation $*$ is the element-wise multiplication (each element in $\boldsymbol{B}$ multiplies a channel on $\boldsymbol{S}$). The addition $+$ in this context represents element-wise matrix addition, where each element in $\mathrm{WN}(\mathbf{X}_{\mathrm{f_i}n})$ is obtained by adding the corresponding elements in $\mathbf{X}_{\mathrm{f_i}n}$ and $\boldsymbol{S} * \boldsymbol{B}$ for each channel. The layer applies a weighted noise map to each 2D feature map of shape $W_i \times H_i$. Since $\boldsymbol{B}$ is a learn-able parameter in the layer, the importance of the added noise at each channel is also learned during the training.

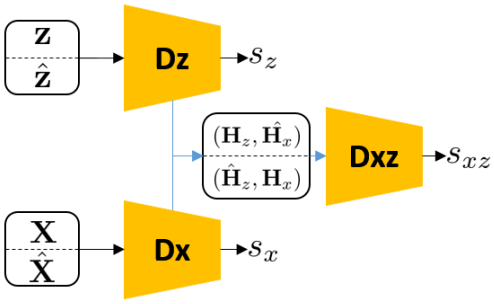


Figure IV.8. Schema for joint discriminators. D_x is a class-projection and patch discriminator for the images, D_z is a dense ResNet for the latent space discrimination and D_{xz} is the joint classification of latent space and images.

IV.3.2 . Class Projection Discriminator

The main discriminator D_x is a patch discriminator architecture [163] combined with the projection discriminator inspired by [151, 164]. D_x starts with the patch discriminator structure to end up with a 2D activation map in the output of size $(p_d, p_d, 1)$ (instead of scalar activation). The $2D$ activation map classifies the different regions of $\mathbf{X}$ by modeling the input image as a Markov random field and assuming independence between pixels separated by more than a patch diameter. Additionally, the mean of the 2D activation map P_x is used to feed the class projection layer as follows,

$$ClassProjection(\mathbf{c}, \mathbf{H}_{\mathrm{x}}) = \mathbf{H}_{\mathrm{x}} * Embedding(\boldsymbol{c}) + P_x, \tag{IV.7}$$

where $\mathbf{H}_x$ is the second to last extracted feature of D_x. The class projection is supported by a $Embedding$ layer that tends to separate the important extracted features at $\mathbf{H}_x$ depending on the class. This term is then added to the patch discriminator output. The class projection operation is intended to help the discriminator to understand the important features for a given class. The embedding acts as a weight for $\mathbf{H}_x$ to correctly classify $\mathbf{X}$ regarding to it class. At the end of the pipeline, D_x is not only able to say if an image comes from $\mathcal{X}$ or not, but also to say if it belongs to the right class. This discriminator is used to train G, who used a similar concept with its class-conditional layers: the generator learns how to leverage the conditioning to created an image close to the seen in $\mathcal{X}$ and it also learns how to change an input class to another so the discriminator does not recognize the original class anymore.

For the latent space discriminator D_z, a dense ResNet layer architecture was used. The second to last activation vector $\mathbf{H}_z$ of this feed-forward network is used to feed D_{zx}. In the same way, the activation map $\mathbf{H}_x$ is passed trough a convolution ResNet block similar to the blocks in F, then trough a dense layer, to be concatenated to $\mathbf{H}_z$. Both activation map and vector are the input for D_{zx}. A schematic representation of this arrangement is shown in Fig. IV.8.

These discriminators are used to classify between 'real' and 'fake' samples. In other words, between the samples coming from the generators and the samples coming from the data set $\mathcal{D}$.

D_x is a sequence of n_{D_x} convolution layers with IN and LeakyReLU activation functions. The SA layer is set after the fist convolution layer. The class embedding layer is plugged at the end to inject the class information. Regarding to D_{xz} and D_z, they have a sequence of $n_{D_{xz}}$ and n_{D_z} dense ResNet layers followed by a simple dense layer at the end of each discriminator with ReLU activation functions and dropout only in the second to last layer (last ResNet layer). The dense ResNet blocks follow the same structure than in Fig. IV.3-b but have dense layers instead of convolution layers (no spatial correlation represented in the latent vector). ReLU activation are also used in the dense ResNet block. Dropout is added for all layers except the layers at the ResNet connections. No activation normalization (batch normalization, instance normalization, etc.), no zero padding, nor down sampling are used in the dense ResNet blocks.

IV.3.3 . Training the adversarial auto-encoder

In this subsection, we analyze the losses and the algorithm implemented to train the adversarial framework. The adversarial losses are based in the hinge loss (Eq. IV.9), usually implemented in kernel machines and successfully applied in the GAN framework, as a score for samples from the original data set (real samples) and the samples coming from the generators (fake samples). The way how F, G, D_x, D_{xz} and D_z are updated by these losses is shown in the Algorithm IV.1. The details on how the generated data are sampled from the F, G is exposed in Algorithm IV.2. The adversarial losses are computed from the output scores from D_x, D_{xz} and D_z accordingly to (IV.8).

More into details, the loss function for the discriminator D_x (Eq. (IV.8)) takes generated samples $\hat{\mathbf{X}}$ from G that follows a posteriori distribution p_G, when $\mathbf{z}$ is sampled from the impose a priori distribution $p_z = \mathcal{N}(\mathbf{0}, \mathbf{I})$, this is $\hat{\mathbf{X}} = G(F(\mathbf{z}), \mathbf{c})$. Since G requires a class input, $\mathbf{c}$ is randomly sampled from a Bernoulli distribution with $p = 0.5$ to promote a good generation for both classes, like in [165]. The discriminator D_z takes $\hat{\mathbf{z}} = F(\mathbf{X})$, where $\mathbf{X}$ represent a sample from the data set $\mathcal{X}$, that follow the p_x original data distribution. It learns the to discriminate $\mathbf{z}$ samples coming from p_z and a posterior samples $\hat{\mathbf{z}}$. For D_{xz} discriminator, couples of $(\mathbf{X}, \hat{\mathbf{z}})$ and $(\hat{\mathbf{X}}, \mathbf{z})$ are sampled and discriminated. This discriminator should differentiate the joint distribution from both couples so when the $\mathbf{X}$ and $\hat{\mathbf{z}}$ happen, is not probable to observer $\hat{\mathbf{X}}$ and $\mathbf{z}$ to happen.

The counterpart of the discriminator losses are shown in Eq. (IV.9), where the hinge loss is used in the opposite way. The objective is that F and G learn from the discriminators losses. When this loss is minimized, the generated samples $\hat{\mathbf{X}}$ and $\hat{\mathbf{z}}$ get the same score that $\mathbf{X}$ and $\mathbf{z}$, so the generators have learned to generate correctly the original data distributions.

$$\begin{aligned} L_{D_x} &= \tfrac{1}{2} E_{\hat{\mathbf{X}}\sim p_G}[\max(1 + D_x(\hat{\mathbf{X}}, \mathbf{c}), 0)] + \tfrac{1}{2} E_{\mathbf{X}\sim p_x}[\max(1 - D_x(\mathbf{X}, \mathbf{c}), 0)] \\ L_{D_z} &= \tfrac{1}{2} \left\{ E_{\hat{\mathbf{z}}\sim p_F}[\max(1 + D_z(\hat{\mathbf{z}}), 0)] + E_{\mathbf{z}\sim p_z}[\max(1 - D_z(\mathbf{z}), 0)] \right\} \\ L_{D_{xz}} &= \tfrac{1}{2} E_{\mathbf{X}\sim p_x;\hat{\mathbf{z}}\sim p_F}[\max(1 - D_{xz}(\mathbf{X}, \hat{\mathbf{z}}), 0)] + \tfrac{1}{2} E_{\hat{\mathbf{X}}\sim p_G;\mathbf{z}\sim p_z}[\max(1 + D_{xz}(\hat{\mathbf{X}}, \mathbf{z}), 0)] \end{aligned} \tag{IV.8}$$

$$\begin{aligned} L_G &= E_{\hat{\mathbf{X}}\sim p_G}[\max(1 - D_x(\hat{\mathbf{X}}, \mathbf{c}), 0)] \\ L_F &= E_{\hat{\mathbf{z}}\sim p_F}[\max(1 - D_z(\hat{\mathbf{z}}), 0)] \\ L_{F,G} &= \tfrac{1}{2} E_{\mathbf{X}\sim p_x;\hat{\mathbf{z}}\sim p_F}[\max(1 + D_{xz}(\mathbf{X}, \hat{\mathbf{z}}), 0)] + \tfrac{1}{2} E_{\hat{\mathbf{X}}\sim p_G;\mathbf{z}\sim p_z}[\max(1 - D_{xz}(\hat{\mathbf{X}}, \mathbf{z}), 0)] \end{aligned} \tag{IV.9}$$

Additionally to the adversarial losses, the reconstruction losses in Eq. (IV.10) were added to enhance the convergence rate of the network and to assure the reconstruction for $\mathbf{z}_r = F(G(\mathbf{z}, \mathbf{c}))$ and $\mathbf{z}$; and $\mathbf{X}_r = G(F(\mathbf{z}))$ and $\mathbf{X}$. The Focal Frequency Loss (FFL) [148] was added to L_r to help the generator to better synthesis the high-frequency details in the images. L_r is weighted by $\lambda > 0$ to help the convergence of the adversarial framework and as a "healthy" initialization of the training. The objective for the generators is also to have a good cycle reconstruction accuracy such as in [166]. Toward this end, a cycle-consistency during the generation is assured by

$$L_r = E_{\mathbf{X}\sim p_x}[L1(\mathbf{X} - \mathbf{X}_r)] + E_{\mathbf{X}\sim p_x}[FFL(\mathbf{X} - \mathbf{X}_r)] + E_{\mathbf{z}\sim p_z}[L1(\mathbf{z} - \mathbf{z}_r)]. \tag{IV.10}$$

Furthermore,a supplementary loss (Eq. (IV.11)) is added for $\mathbf{X}_r$ generation. The objective of this loss is to use the power of the discriminator D_x to ensure a good quality in the reconstruction. This loss is complementary to L_r and intended to observe the details in reconstruction quality that can not be observed by $L1$ or FFL.

$$L_{D_{x_r}} = E_{\mathbf{X}\sim p_x}[\max(1 - D_x(\mathbf{X}_r, \mathbf{c}), 0)]. \tag{IV.11}$$

To force the G to properly generate each class, particularly when a "class switching" is produced by F and G, an adversarial loss is proposed to train the set of networks (Eq. (IV.12)). Here, the projector discriminators acts as a class-conditioned discriminator to say if the generation $\mathbf{X}_{cs} = G(F(\mathbf{X}, \mathbf{c}_s))$ belongs to p_x, where $\mathbf{c}_s$ is the opposite of the original class for $\mathbf{X}$ in the one-hot embedding for $\mathbf{c}$.

$L_{D_{x_{cs}}}$ is used to update the D_x weights while $L_{G_{X_{cs}}}$ updates the generators weights, as shown in Algorithm IV.1.

$$\begin{aligned} L_{D_{x_{cs}}} &= E_{\mathbf{X}\sim p_x}[\max(1 + D_x(\mathbf{X}_{cs}, \mathbf{c}_s), 0)], \\ L_{G_{x_{cs}}} &= E_{\mathbf{X}\sim p_x}[\max(1 - D_x(\mathbf{X}_{cs}, \mathbf{c}_s), 0)], \end{aligned} \tag{IV.12}$$

The losses in Eq. (IV.12) work similarly to the informative loss proposed by [165], since the mutual information between $\mathbf{c}$ and $\mathbf{z}$ is maximized by the bias of the class-projection discriminator D_x.

Algorithm IV.1 Adversarial AutoEncoder training algorithm.

for number of iterations **do**
  for number of batches **do**
  ▷ Sampling from dataset and distributions.
    $\mathbf{z} \sim \mathcal{N}(\mathbf{I}, \mathbf{0})$; $\mathbf{n} \sim \mathcal{N}(\epsilon, \mathbf{0})$; $\mathbf{X} \sim p_x$.
    $\mathbf{c} \leftarrow$ class for $\mathbf{X}$.
    Freeze F,G weights. Defreeze Dxz weights.
    Forward network call (see Algorithm 2)
    Compute Dxz losses (see Eqs. on Section 4.3)
  ▷ Compute gradients and update weights
    $\theta_{D_{xz}} \leftarrow \nabla_{\theta_{D_{xz}}}(L_{D_x} + L_{D_{xz}} + L_{D_z} + L_{D_{x_{cs}}})$
    Frozen Dxz weights. Defrozen F,G.
    Forward network call (see Algorithm 2)
    Compute F,G losses (see Eqs. on Section 4.3)
  ▷ Compute gradients and update weights
    $\theta_G \leftarrow \nabla_{\theta_G}(L_G + L_{F,G} + \lambda L_r + L_{G_{x_{cs}}} + L_{D_{x_r}})$
    $\theta_F \leftarrow \nabla_{\theta_F}(L_F + L_{F,G} + \lambda L_r + L_{G_{x_{cs}}} + L_{D_{x_r}})$
  end for
end for

Algorithm IV.2 Forward network subsection called from Algorithm 1.

▷ Marginal F and G sampling.
$\hat{\mathbf{X}} \leftarrow G(\mathbf{z}, \mathbf{n}, \mathbf{c})$.
$\hat{\mathbf{z}} \leftarrow F(\mathbf{X})$.
▷ Cycle F and G generation.
$\mathbf{X}_r \leftarrow (G \circ F)(\mathbf{X}, \mathbf{n}, \mathbf{c})$.
$\mathbf{z}_r \leftarrow F(\mathbf{X}_r)$.
▷ Class switching (cw).
$\mathbf{X}_{cw} \leftarrow (G \circ F)(\mathbf{X}, \mathbf{n}, \mathbf{c}_s)$.

IV.3.4 . Remarks on the image reconstruction quality

The proposed generator was conceived to generate high-quality images to get as close as possible to the data set images in probabilistic terms. For this end, G architecture was adapted to improve the image reconstruction. This task is assured in terms of loss by the FFL and the $L1$ loss.

At the same time, the discriminators play an important role in the image quality. D_x must be able to see the details between the generated samples ($\hat{\mathbf{X}}$, $\mathbf{X}_{cs}$and $\mathbf{X}_r$) and the original data $\mathbf{X}$. This is assured by the patch discriminator architecture adopted and the SA layer. Most of the added layers in this sense were designed based on the cGAAE architecture on the observed final reconstruction quality.

The combination of proposed losses and particular layer allowed to achieve a good reconstruction quality. It is worth mentioning that part of the improvement in the high-frequency details comes from empirical tests on the placement of the WN layers. The ResNet connections brought also an improvement during the tests in terms of reconstruction. The $\mathbf{z}$-skip connections and the class conditioning at different levels of synthesis resolution on G improves not only the convergence but also in the reconstruction. The key idea in the $\mathbf{z}$-skip connection is to avoid passing all the important information for the image synthesis at the very beginning of the generator (e.g., the first dense layer at G). Instead, the conditional layers decide where the latent information (including the class label) is relevant to the generator to better reconstruct the image and to satisfy the adversarial losses.

IV.3.5 . Remark on the training stability

Generative neural network stability has been a topic of interest since the first application of this method. Several problems such vanishing gradient, mode collapse and under-fitting are common challenges for this type of NN. In this work, several stability techniques were implemented to ensure an stable training. This techniques brought a faster training while reducing the impacts of the modifications of architecture or hyper-parameters changes in its stability.

The hinge loss was the more stable tested loss for the adversarial game, particularly in terms of the vanishing gradient issue. The Two Times-scale Update Rule (TTUR) [167] with Adam optimization showed a great improvement in the convergence rate during the training when decoupling the discriminators and generators learning rates. The hyper-parameters choice becomes simpler with this learning rate choice rule. The generators and discriminators may be considered as players in a game where they look for a Nash equilibrium. Thanks to TTUR, a stationary local Nash equilibrium is assured at most of the training epochs. In this case, the potential of the players becomes a secondary problem for the stability of the training. In other words, the discriminator capacity to classify $\mathbf{X}$ versus $\hat{\mathbf{X}}$ regarding the generator potential to create good $\hat{\mathbf{X}}$ samples is not the more relevant aspect in the stability. Instead, it is enough to ensure that both the discriminator and generator are sufficiently expressive to capture or generate all the information from $\mathbf{X}$. This means, that even if one of the players is not powerful enough, the adversarial game will be stable and the losses will accuse who is the problematic player. This brings substantial simpleness in terms of the architecture development, reducing the task of adding or changing layers or modifying losses.

In [151], authors demonstrated in an empirical way that, the F and G can be decoupled in terms of learning rate, similarly to TTUR, they make the F to converge to a local stable solution before making G to try to synthesize $\mathbf{X}$ from $\mathbf{z}$. To clarify, the idea is to increase by 10 times the learning rate of F regarding to the learning rate of G so the space $\mathcal{Z}$ is a fair representation of $\mathcal{X}$ when G is training itself.

Together with the hinge loss and the TTUR technique, the Spectral Normalization (SN) also contributes to the stability during the training. The largest singular value of the NN's weights of each layer is used as normalizing value after each weights updating. The SN is one of the less constraining technique to encourages Lipschitz continuity, which is a desired property for the discriminators to improve the stability and reduce the possibility of mode collapse [151, 168, 169]. The Lipschitz constraint is suitable when losses like Wasserstein or hinge one are used for the discriminator. These losses give a score with values from $-\infty$ to ∞ for the different inputs, so they can easily diverge if the discriminator weights are not regularized. Furthermore, the SN also shows to boost the generators performance when it is applied to its weights [170]. That is, G is not ill-conditioned at the beginning and the SN "re-initialization" of weighs at each epochs improves the stability of the training and avoids the mode collapse.

IV.4 . Numerical results

The generative framework was trained over a data set with $M = 324$ simulated UT M-TFM images representative of a complex geometry, where $C_{modes} = 9$ modes have been reconstructed in a region of interest of size $W = H = 128$. The experimental acquisitions were also done on the same inspection

problem. Thus, the same amount of images than the simulated one were produced from a mock-up. More information on the parameters for the simulation and the experience as well as the sampled employed can be found in Chapter III. All images were normalized by modes. The global max-min of each reconstruction mode was used to set each image mode group in the range of $[0.0, 1.0]$.

The multi-channel images are arranged in single-channel images for the training to be treated as independent $\mathbf{X}_n$ samples, so $C = 1$ and $N = 324 \times 9 \times 2 = 5832$. This choice can be seen as a data augmentation strategy for the class-switch task. Indeed, the generators do not need to learn the correlation between modes but they focus on the gap between the classes. The new data set arrangement allow to create a vector of parameter $\mathbf{p}_n$ for each image $\mathbf{X}_n$ with $n_z = 5$. This vector contains labels of the flaw geometry and position (L and β), the transverse UT wave celerity (c_T), the back-wall angle (α) and the M-TFM reconstruction mode (M). The experimental measurements were done on 4 different flaws represented at different c_T and α values.

Regarding the hyper-parameters of the generative cGAAE, the latent code is fixed to $d = 96$ with a number partition $s = 4$, resulting in $\dim(z_i) = 24$. The vector ϵ to sample $\mathbf{n}$ noise is set to 0.1. The framework was trained following the TTUR with an Adam optimization, the learning rate is 0.68×10^{-3} for F and D, and 0.1×10^{-3} for G. Adam's hyper-parameters β_1 and β_2 are fixed to 0.5 and 0.999, respectively. The batch size per iteration is fixed to 64. The dimension of embedding for $\mathbf{c}$ is set to 2 for G input, while for the D_x is set to 64. ϵ_{IN} is set to 1×10^{-3}.

.D_x counts with $n_{D_x} = 4$ blocks of the layer sequences described in IV.3.2. Regarding to D_{xz} and D_z, the number of neurons at the dense layers is 32 and 64, respectively. The dropout is set to 0.2. The number of ResNet blocks is $n_{D_{xz}} = n_{D_z} = 6$. The patch is set to $p_d = 14$. Reconstruction loss weight is set to $\lambda = 2 \cdot 10^4$.

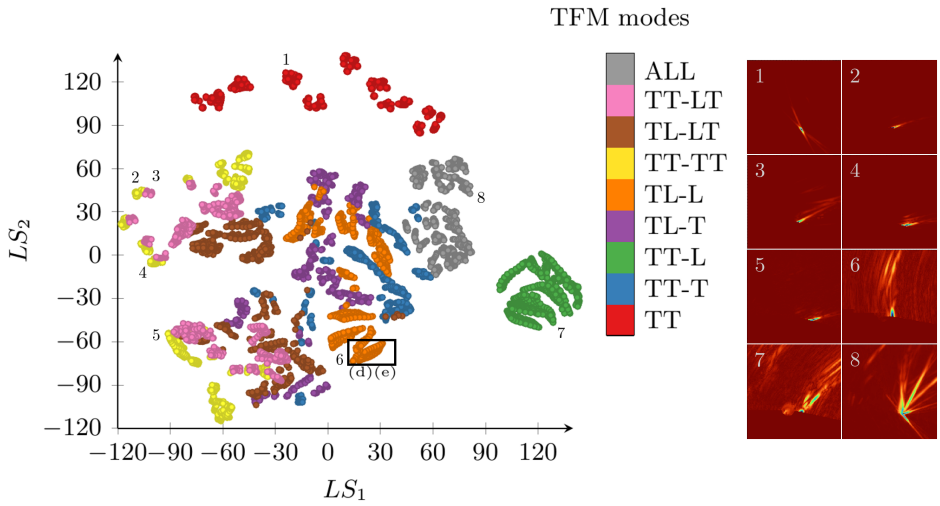
The model is trained over $\sim 11\,\mathrm{k}$ epochs on $\mathcal{D}$ partitioned as follows: one of the four flaw geometries is kept out during the training (3mm, vertical). This flaw's images constituted the test set. A validation set is created with the rest of the data by splitting it into the 75% of the remaining images. The criteria for the validation set creation is to take half of the fidelities couples from the image samples. This is, the validation set contains couples never seen during the training. As a result, only $\sim 2\,\mathrm{k}$ samples are left for training set. The optimization is performed on a 4 GPU units Nvidia V-100 with 25 GB of RAM each. A stop criteria is set in the validation reconstruction loss to stop the training by an early stopping.

We analyze the obtained results accordingly to two parts, the first part is the realistic data generation from the couples F and G is in focus, where the latent space is explored to show the generative potential. Secondly, an inverse problem is assessed by using the generated data set to show the performance improvement when the present generative framework is applied.

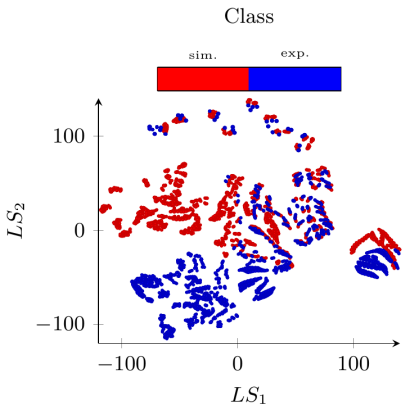
IV.4.1 . Realistic data generation by sampling the generator input

Once trained, F can be forward-propagated by a sample $\mathbf{X}_n$ associated with a fidelity level (or class) $\mathbf{c}_n$ and a set of parameters to get a latent space vector $\mathbf{z}_n$. When all samples in $\mathcal{X}$ are forwarded, the conditional $\mathcal{Z}$ space can be built as it is shown in Fig. IV.9. The shown representation is the result of processing $\mathcal{Z}$ space from the dimension of $d = 96$ to two dimensions with t-SNE algorithm [70]. The resulting plot show how each sample is coded by F and how some of the parameters impact its latent representation.

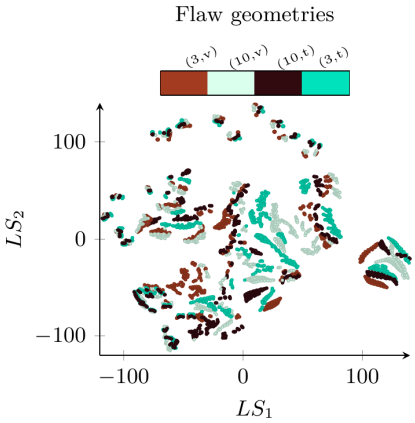
The latent space structure in Fig. IV.9 exposes a hierarchy in $\mathcal{Z}$. This hierarchy may be compared to data set $\mathcal{X}$ intrinsic relations between the parameters $\mathbf{p}$ and images. For instance, the shape of the echos and the background noise texture is different for each modes, so it is expected to find the principal clusters in term of mode. This is illustrated by the scatter plot together with the zoomed flaw images near to some clusters in Fig. IV.9(a). The classes present tow main clusters in $\mathcal{Z}$ with a similar structure, since the two fidelity levels in $\mathcal{X}$ are affected in a similar way by $\mathbf{p}$ (Fig. IV.9(b)). Regarding the flaw geometries, is natural to find secondary clusters since the signature (i.e., the echo) in a M-TFM image can be considered as a total different class per each geometry. Each flaw geometry secondary cluster in $\mathcal{Z}$ represents the variation of reconstruction parameters (c_T and α) in $\mathcal{X}$, as shown in Fig. IV.9(d) and (c). Since each parameter modifies the image in a different way (mainly shift for c_T and rotations for α)



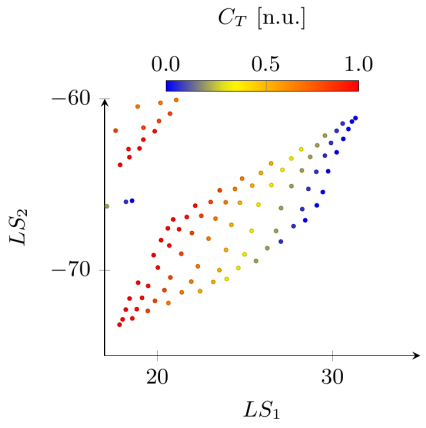
(a) Whole data set, colored by modes (left). Some images linked to the enumerated points are shown at the right.



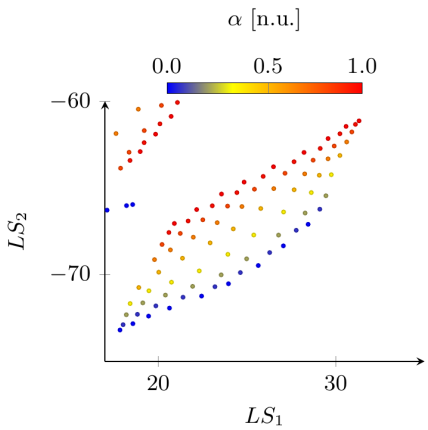
(b) Whole data set, colored by fidelity levels (classes)



(c) Whole data set, colored by flaw geometry parameters

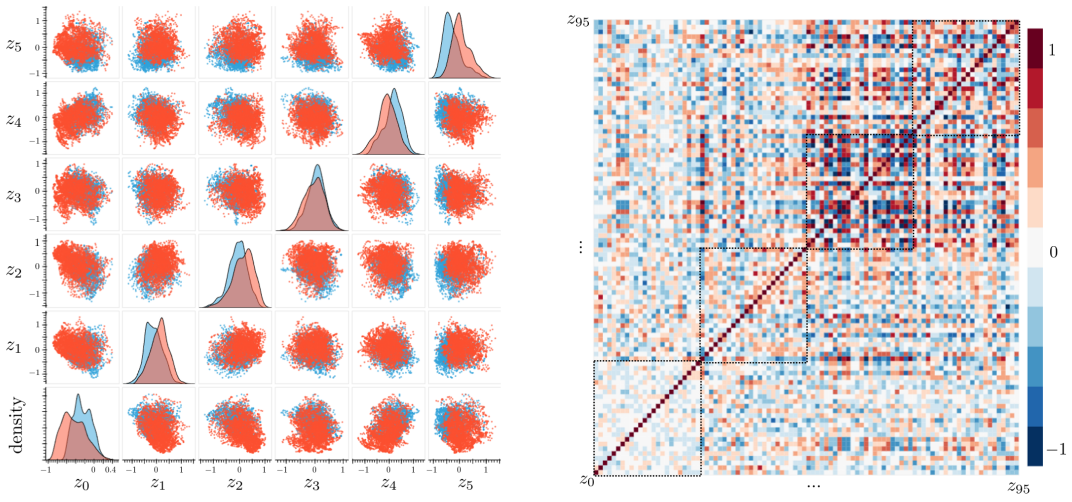


(d) Zoom from rectangle in (a), colored by wave velocity velocity



(e) Zoom from rectangle in (a), colored by back-wall slope angle

Figure IV.9. Representation of latent space $\mathcal{Z}$ in a t-SNE 2D projection. The complete data set $\mathcal{X}$ is presented, colored by the parameters and class from the simulation and mock-up configuration. LS: Latent Space coordinates for a 2D t-SNE embedding. References for flaw geometries coloring: $(3, \mathrm{t})$ is the 3 mm tilted flaw geometry, $(10, \mathrm{v})$ is the 10 mm vertical flaw geometry, and so on. (a) shows the clustering by modes. (b) shows the clustering by classes with some TFM images samples plotted near the cluster where they belong (the images are zoomed to show better the echos shape). (c) shows the clustering by flaw geometries. (d) and (e) is the a selected cluster (black rectangle in (a)) colored by two of the parameters: c_T and α.



(a) (b)

Figure IV.10. Latent space distributions for the first 6 components (a) and correlation matrix for all components (b). The dotted squares demarcate the 4 $\mathbf{z}$ partitions.

it is expected to find two principal directions that are weakly correlated. The figure shows the zoom on one of the cluster, and each cluster presents a different local structure depending to the others parameters in $\mathbf{p}$.

Accordingly to the distribution of $\mathcal{Z}$ and the correlation matrix as is shown in Fig. IV.10, we observe that the latent space respect the imposed distribution during the training. We also observe that the correlation matrix demonstrates that there is a hierarchy in $\mathcal{Z}$. The $\mathbf{z}$-partitions contain more relevant information in the first two stages, while in the last two, $\mathbf{z}$ is not informative at all. It is important to mention that $\mathcal{Z}$ structure depends on the data set a priori distribution, the imposed distribution learned by D_z, and G architecture definition. For the last point, this is closely linked to how G forward $\mathbf{z}$ to retrieve the reconstruction of $\mathbf{X}$ and the correct $\mathbf{X}_{cs}$.

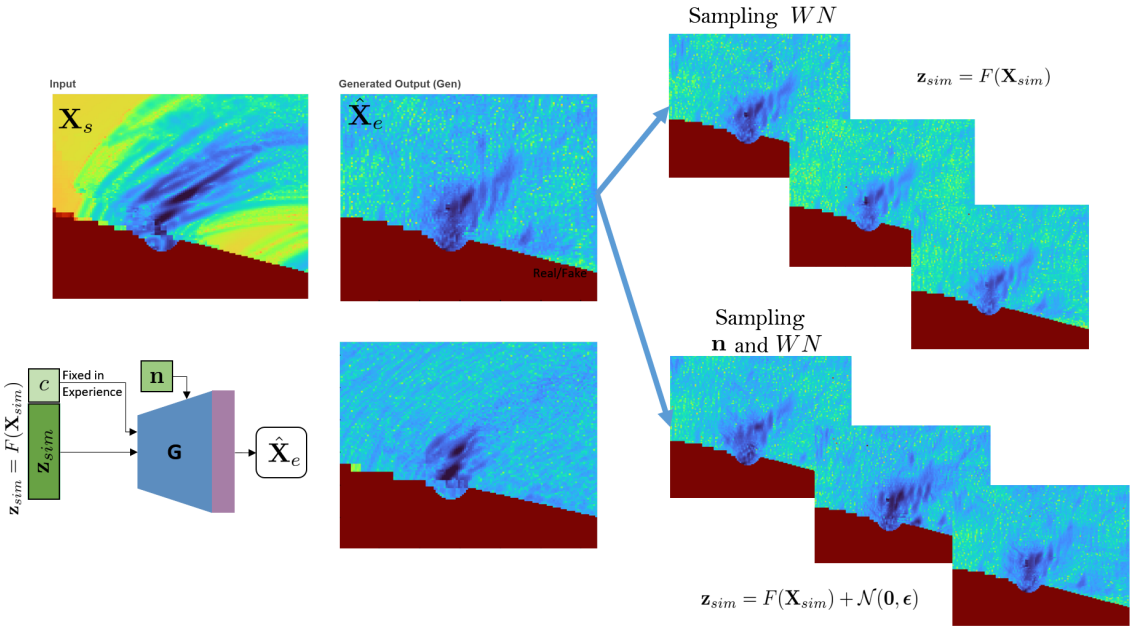
This correlation matrix hierarchy can be explained from how G uses $\mathbf{z}$ to synthesize an image. G generates the most important features of $\mathbf{X}$ from $\mathbf{z}_0$ at the first layers (Fig. IV.4). That is why $\mathbf{z}_0$ must be the most informative of the 4 partitions. Subsequently, G drives the content generation and the class switching after the ResNet block, more specifically in the cST. This layer is conditioned by $\mathbf{z}_1$, so this partition is relevant too for the objectives of the losses, particularly for the class switching objective. Finally, the last two ResNet blocks add the last details to $\mathbf{X}$, mostly the background noise of the image and the high-frequency features. Here the WN layers are the most relevant source of noise, giving a stochastic input for the synthesis of the final details of $\mathbf{X}$, so $\mathbf{z}$ becomes useless at this stage. Consequently, the last $\mathbf{z}$ partitions are less informative.

To discuss the $\mathbf{X}$ and $\mathbf{X}_{cs}$ generation, we show in Fig. IV.11 some examples of the new "realistic" data set. We use the test set to show the generalization of the framework to learn the gap between two domains: low-fidelity versus high-fidelity level.

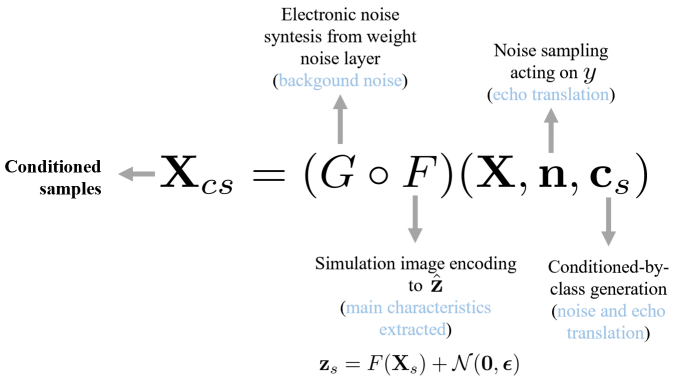
The Structural Similarity Index Measure (SSIM) [171] is used to compare each generated image with the experience ground truth. These metrics assess the similarity in terms of structural and perceptual features and its range is from -1.0 to 1.0, being 1.0 the highest similarity. We quantify the fairness of generation for the non-seen flaw geometry (test set) by a mean of SSIM, given by

$$SSIM_{\text{mean}} = \frac{1}{N_T \cdot S} \sum_{n=1}^{N_T} \sum_{i=1}^{S} SSIM(\mathbf{X}_{\text{exp}_n}, \mathbf{X}_{cs_{n,i}}), \tag{IV.13}$$

where $\mathbf{X}_{\text{exp}_n}$ is an experimental sample in the test set with $N_T = 729$ realizations. $\mathbf{X}_{cs_{n,i}}$ is the class switched prediction from the cGAAE from simulation to experience. In this case, the simulation and



(a)



(b)

Figure IV.11. Generator sampling strategy for the realistic data set. (a) A sample of realistic generated data from a simulated input $\mathbf{X}_{sim}$, with sampling on $\mathbf{n}$ and on WN layers independently. The figure shows how the style (background noise) is separated from the content (echo geometry) by the two stochastic inputs on G. (b) express the sampling in a functional notation while specifying the contribution of the features generated on $\mathbf{X}_{cs}$ by each input of $(G \circ F)$.

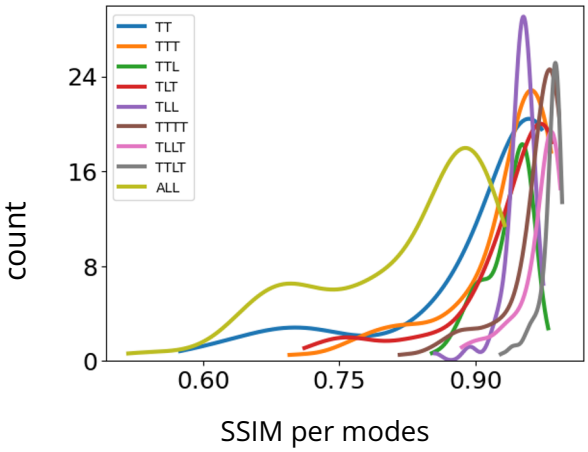


Figure IV.12. SSIM frequency by TFM reconstruction modes (closer to 1.0 is better), given by the kernel density estimation. This metric shows the similarities between the experimental ground truth and the realistic generation given by $G \circ F$ NNs.

experimental from the test set are both produced over the same parameters (flaw geometry, material T celerity and back-wall slope). The sum over i represents the several sampling for $\mathbf{X}_{cs_{n,i}}$, where $S = 20$ following the Fig. IV.11 generation procedure and sampling $\mathbf{n}$ and WN layer at the same time. The reported $SSIM_{\text{mean}}$ is 0.92, showing the generalization power of the generative neural network. A density distribution per TFM reconstruction mode is shown in Fig. IV.12.

In Fig. IV.13 the evolution of loss curves is given in function of the epochs for the reconstruction loss, and the discriminator and generator adversarial losses.

IV.4.2 . Exploitation of realistic generation outcomes applied to inversion tasks

Once the realistic generation was validated for the presented framework, a test on cGAAE data generation is propose by using the generated data in an inversion problem. The objective is to train a regressor (inverse) model $\mathcal{M}^{-1}$ such that

$$\mathcal{M}^{-1} : \mathcal{I} \rightarrow \mathcal{P} \tag{IV.14}$$

where $\mathcal{P}$ corresponds to the set of parameters to be retrieved based on the M-TFM image data set $\mathcal{I}$ provided as input. To analyze the impact of synthetically generated images on the inversion performance, we considered three different data set M-TFM settings referred hereafter as $\mathcal{I}_{SIM}$, $\mathcal{I}_{cGAAE}$ and $\mathcal{I}_{EXP}$. For the purpose of this section, the original simulated data set accounting 324 samples was enlarged f to 5935 samples by uniformly sampling of the $\mathcal{P}$ space accordingly to the procedure provided in Chapter III, in the Fig. III.5. The parameters ranges studied were $2.0 \leq L \leq 12.0$, $-20.0 \leq \beta \leq 0.0$, $10.0 \leq \alpha \leq 18.0$ and $3080.0 \leq c_T \leq 3380.0$.

$\mathcal{I}_{SIM}$ identifies the set of simulated images with dimension is $[5935, 128, 128, 9]$ labeled by $\mathbf{p}_s$. The multi-channel M-TFM images have a dimension of 128×128 pixelsand 9 channels, one per reconstruction mode. $\mathcal{I}_{EXP}$ contains the images generated from the experimental acquisition and labeled by $\mathbf{p}_e$. $\mathcal{I}_{EXP}$ dimension is then $[324, 128, 128, 9]$, coming from the data set $\mathcal{D}$. It is worth to note that for testing purposes a flaw was left out for the cGAAE training and validation the one considered for the inversion task. $\mathcal{D}$ is rearranged from $[M, W, H, C]$ to $[M, W, H, C_{modes}]$ format to create the multi-channel images accordingly to $\mathcal{I}_{SIM}$. Finally, the $\mathcal{I}_{cGAAE}$ data set was created by exploiting F and G generators.

The enhanced realistic images are generated by feeding the $\mathcal{I}_{SIM}$ images in input to our architecture. By sampling $\mathbf{n}$ and the WN layers simultaneously at a rate of $S = 3$ samples per simulation input image (see Fig. IV.11), a multi-fidelity $\mathcal{I}_{cGAAE}$ data set is generated with the dimensions of $[17805, 128, 128, 9]$. That is, this data set contains 3 realistic representation of each simulated image composed by what we have called $\mathbf{X}_{cs}$ images. To respect the original inter-modes correlation for a

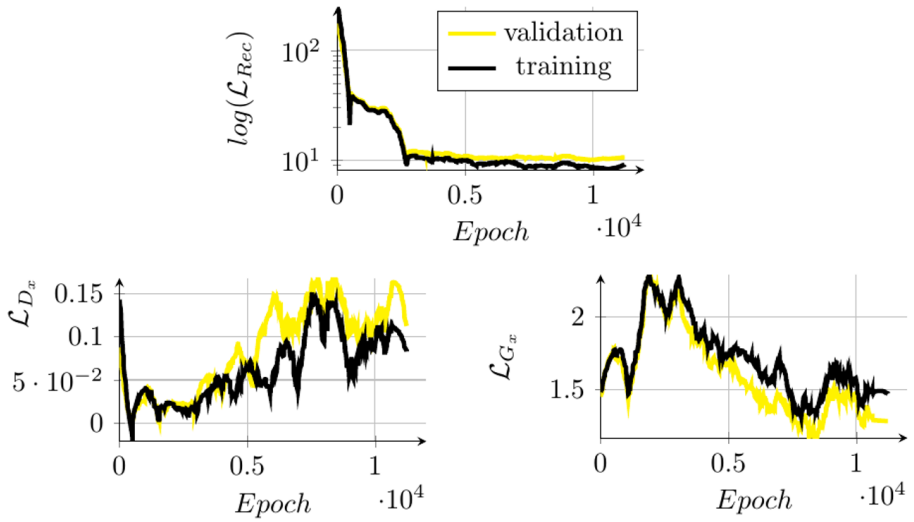


Figure IV.13. Loss evolution versus epochs for AAE training on M-TFM data set. The training loss is in black, and the validation loss is in yellow. $log(\mathcal{L}_{rec})$ is the reconstruction loss for $\mathbf{X}$. $\mathcal{L}_{D_x}$ and $\mathcal{L}_{G_x}$ are two of the adversarial loss. In this case, the hinge loss over $\mathbf{X}$ is shown. $\mathcal{L}_{G_x}$ decreases over the epochs, while $\mathcal{L}_{D_x}$ increases, showing that G is generating images similar to the training data set $\mathcal{X}$ and D_x is less capable of discriminating generated samples. For $\mathcal{L}_{D_x}$ and $\mathcal{L}_{G_x}$, smoothed curves are presented.

given multi-channel M-TFM image, the noise sampling for G is constant for each instance on $\mathcal{I}_{cGAAE}$, so the noise at the 9 channel images represents a single FMC acquisition. To clarify, G output has a single channel image output, so to represent a 9-channel M-TFM image, G must be forwarded 9 times, one per mode. To respect the fact that during an acquisition, the 9 modes are impacted by the same measurement errors and noise, $\mathbf{n}$ must stay fixed when building an image of shape$(128, 128, 9)$. Finally,$\mathcal{I}_{cGAAE}$ images are labeled by the enlarged $\mathbf{p}_s^*$ vector, which is a simple repetition of $\mathbf{p}_s$ to label the new images, by respecting the original $\mathbf{p}_s$ coming from the simulation side, but repeated 3 times for the new realistic samples. $\mathbf{p}_s$ and $\mathbf{p}_s^*$ were normalized by parameter in the range of $[0.0, 1.0]$. Referring to Eq. (IV.14), he described data sets were used for the inversion problem:

$$\mathbf{p} = \mathcal{M}^{-1}(\mathbf{X}), \tag{IV.15}$$

where $\mathcal{M}^{-1}$ is the NN specifically tailored to be applied on M-TFM for regression tasks. $\mathcal{M}^{-1}$ is an in-house inverse deep learning model developed for M-TFM.

The hyper-parameters for the inversion network $\mathcal{M}^{-1}$ are presented in Tab. IV.1. The architecture with Adam optimizer with a learning rate of $1 \cdot 10 - 4$, and a batch size of 32. The patience for the early stopping is 500 epochs. The loss is a Mean Square Error (MAE) applied in a supervised way to the set of parameters.

The first and last layer is modified to match $\mathbf{X}$ and $\mathbf{p}$ dimensions, respectively with the last layer has linear activation. It is worth to mention that the main purpose of the analysis performed in this section is to assess the performance changes due to the use of synthetically generated data for the regression task addressed. The choice of the most suitable NN architecture to fit of the data considered in this section were not considered in the analysis proposed, certainly it would and interesting research axis. The resulting NN ($\mathcal{M}^{-1}$) is trained for the inverse problem for three different cases, first the training data set is $\mathcal{I}_{SIM}$ to train $\mathcal{M}^{-1}$ and later tested on $\mathcal{I}_{EXP}$. In the second case, the enlarged realistic data set $\mathcal{I}_{cGAAE}$ is the training data set and the resulting $\mathcal{M}^{-1}_{cGAAE}$ is also tested on $\mathcal{I}_{EXP}$. Finally, a hybrid (or enhanced) data set $\mathcal{I}_{SIM+cGAAE}$ is used to train $\mathcal{M}^{-1}_{SIM+cGAAE}$, where the simulated data and realistic

Table IV.1. CNN Architecture used for $\mathcal{M}^{-1}$

Layer	Type	Output Size	Kernel Size	Activation
0	Input	$128 \times 128 \times 9$	-	-
1	Conv2D	$128 \times 128 \times 16$	3×3	ReLU
2	Conv2D		3×3	ReLU
3	Conv2D		3×3	ReLU
4	Conv2D MaxPooling2D	$128 \times 128 \times 16$ $64 \times 64 \times 16$	3×3 2×2	ReLU
5	Conv2D	$64 \times 64 \times 32$	3×3	ReLU
6	Conv2D		3×3	ReLU
7	Conv2D		3×3	ReLU
8	Conv2D MaxPooling2D	$64 \times 64 \times 32$ $32 \times 32 \times 32$	3×3 2×2	ReLU
9	Conv2D	$32 \times 32 \times 64$	3×3	ReLU
10	Conv2D		3×3	ReLU
11	Conv2D		3×3	ReLU
12	Conv2D MaxPooling2D	$32 \times 32 \times 64$ $16 \times 16 \times 64$	3×3 2×2	ReLU
13	Flatten	16384	-	-
14	Dropout (0.33)	16384	-	-
15	Dense	512	16384×512	ReLU
16	Dense	256	512×256	ReLU
17	Dense	4	256×4	Linear

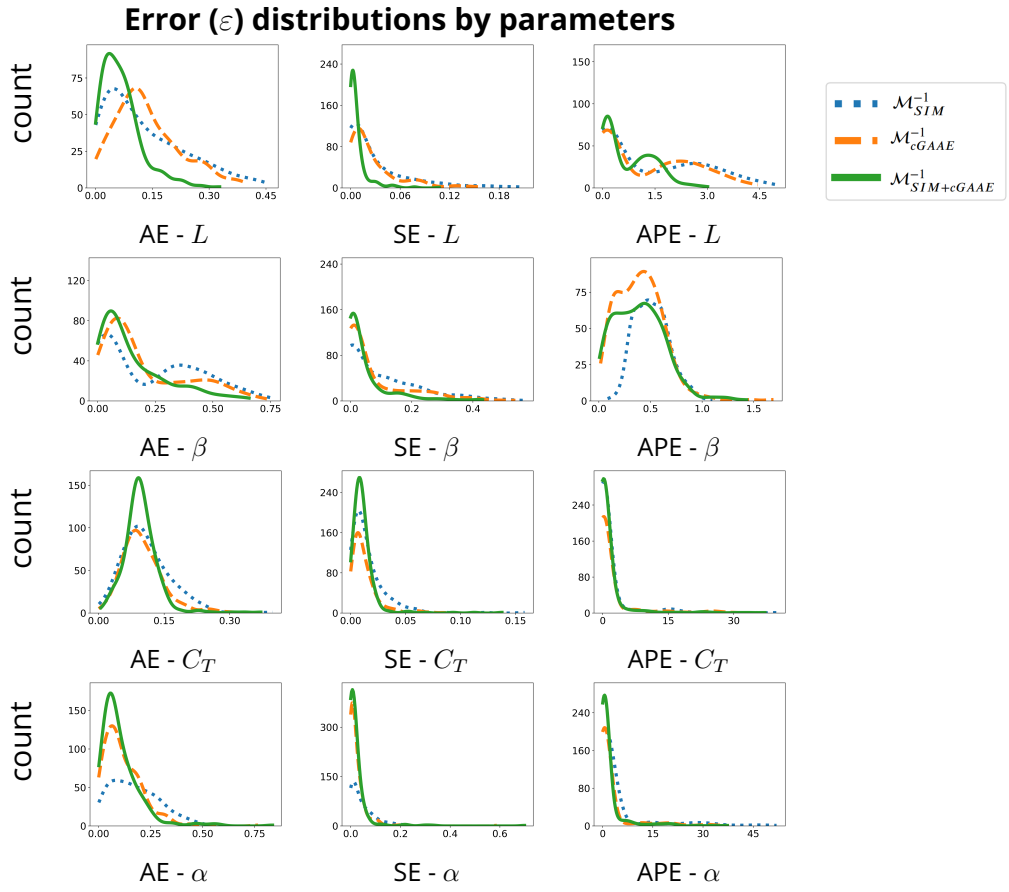


Figure IV.14. Results of inverse problem training. $\mathcal{M}^{-1}_{SIM}$, $\mathcal{M}^{-1}_{cGAAE}$ and $\mathcal{M}^{-1}_{SIM+cGAAE}$ performances for $\mathcal{I}_{EXP}$ data set (324 instances). Kernel density estimation is implemented to show the probability density estimation for $\mathcal{M}^{-1}$ prediction error frequency over the $\mathcal{I}_{EXP}$ instances. The measured magnitude in x-axis is a metric applied to the difference between $\mathbf{p}_{true}$ and $\mathbf{p}_{predicted} = \mathcal{M}^{-1}(\mathbf{X})$, such as Absolute Error (AE), Squared Error (SE) and Absolute Percentage Error (APE). The y-axis is the instance frequency for each error value.

data are put together in a single data set. The objective of this study is to show the differences in the performance when realistic data are used in an inverse problem, instead of just using just $\mathcal{I}_{SIM}$.

The neural network is trained with the same criteria regardless of the data set fed to the network. A validation set of 20% of samples at each $\mathcal{I}$ set is kept out to prevent over-fitting, an early-stopping is set for each training with a patience of 500 epochs, a learning rate of 10^{-4} on an Adam optimizer. Each model is fitted in different fidelity levels and the performances are compared here. The training is done on 4 GPU units Nvidia V-100 with 25 GB of RAM each.

The Fig. IV.14 shows an improvement when the predictor has seen realistic data, and not only simulated data. One can notice that the green curves present a smaller dispersion and a smaller error mean. Additionally, it can be noticed that $\mathcal{M}^{-1}_{SIM+cGAAE}$ distributions in green present the highest value, particularly for the inversion of L. A numerical assessment for this representation for different metrics is exposed in Tab. IV.2, were we observe that $\mathcal{M}^{-1}_{cGAAE}$ and $\mathcal{M}^{-1}_{SIM+cGAAE}$ over-perform the task compared to $\mathcal{M}^{-1}_{SIM}$.

Looking at the reported error in Tab. IV.2, one can notice that the inversion of L reports the best performance in general when $\mathcal{M}^{-1}_{SIM+GAN}$ is used to predict experience data. In terms of improvements, $\mathcal{M}^{-1}_{SIM+cGAAE}$ reported an improvement of 26% in EV metric and $22,7\%$ in R2 metric for α, compared to $\mathcal{M}^{-1}_{SIM}$. Furthermore, the model trained on the enhanced training set, is shown to be capable of explaining only around half of the variation for α, while training the same model on simulated and

Performance for parameter L. Test on $\mathcal{I}_{EXP}$.

	$\mathcal{M}^{-1}_{SIM}$	$\mathcal{M}^{-1}_{cGAAE}$	$\mathcal{M}^{-1}_{SIM+cGAAE}$
EV (↑)	0.811	0.813	**0.938**
MAE (↓)	0.131	0.138	**0.069**
RMSE (↓)	0.171	0.163	**0.087**
R2 (↑)	0.761	0.784	**0.938**
MAPE (↓)	1.394	1.298	**0.744**

Performance for parameter β. Test on $\mathcal{I}_{EXP}$.

	$\mathcal{M}^{-1}_{SIM}$	$\mathcal{M}^{-1}_{cGAAE}$	$\mathcal{M}^{-1}_{SIM+cGAAE}$
EV (↑)	0.602	0.697	**0.791**
MAE (↓)	0.233	0.201	**0.152**
RMSE (↓)	0.313	0.271	**0.213**
R2 (↑)	0.201	0.401	**0.63**
MAPE (↓)	0.512	0.399	**0.393**

Performance for parameter c_T. Test on $\mathcal{I}_{EXP}$.

	$\mathcal{M}^{-1}_{SIM}$	$\mathcal{M}^{-1}_{cGAAE}$	$\mathcal{M}^{-1}_{SIM+cGAAE}$
EV (↑)	0.917	0.924	**0.946**
MAE (↓)	0.109	0.101	**0.097**
RMSE (↓)	0.123	0.112	**0.105**
R2 (↑)	0.854	0.879	**0.893**
MAPE (↓)	1.665	1.859	**1.347**

Performance for parameter α. Test on $\mathcal{I}_{EXP}$.

	$\mathcal{M}^{-1}_{SIM}$	$\mathcal{M}^{-1}_{cGAAE}$	$\mathcal{M}^{-1}_{SIM+cGAAE}$
EV (↑)	0.594	0.799	**0.809**
MAE (↓)	0.174	0.117	**0.104**
RMSE (↓)	0.21	0.149	**0.143**
R2 (↑)	0.577	0.786	**0.804**
MAPE (↓)	3.61	2.550	**1.906**

Table IV.2. Metrics for $\mathcal{M}^{-1}_{SIM}$, $\mathcal{M}^{-1}_{cGAAE}$, and $\mathcal{M}^{-1}_{SIM+cGAAE}$ tested on $\mathcal{I}_{EXP}$ data set. Each table represents the prediction of a parameter for all $\mathcal{M}^{-1}$ inversion problems. The metrics are computed over a normalized space $\mathcal{P}$ on the range of $[0.0, 1.0]$. (↑) and (↓) symbols inform when the metric is better when is higher or is better to be lower, respectively. The best metrics are in **bold**. EV: Explained variance. MAE: Mean Absolute Error. RMSE: Rooted Mean Squared Error. R2: R-squared error. MAPE: Mean Absolute Percentage Error.

realistic data coming from the cGAAE results on around the 80% of the explained variation for the same $\mathcal{M}^{-1}$ inversion model. These improvements underline the potentiality beyond the used of an enhanced training set to fit a more robust $\mathcal{M}^{-1}$ than the one that can be obtained from simulated data only.

IV.5 . Discussion

To find a convenient generative architecture, many options were explored before the final cGAAE architecture definition. One of the most difficult issues to address for the development of the architecture was how to use $\mathbf{c}$ as an input for fidelity level switching meanwhile avoiding model instabilities in the training phase. Toward this end, we tailored the design of the deep neural network accordingly to the choices introduced in Section III.2 build G with an input $\mathbf{c}$. We tailor the architecture based on the assumption that $\mathbf{c}$ switching can be achieved by a set of transformations assured by the layers cST and AdaIN. In other words, the differences between fidelity levels exist in content and style, so the cGAAE must contain these transformations to perform the task of generating $\mathbf{X}_{cs}$. Including $\mathbf{c}$ at each resolution stage on G gave the cGAAE the possibility to use the class (or fidelity level) information better. Adding $L_{D_{x_{cs}}}$ and $L_{G_{x_{cs}}}$ are also part of the propose solution for this problem.

Furthermore, the presented architecture has been obtained after an extensive evaluation of promising state-of-the art architectures that tent to solve the problem of using $\boldsymbol{c}$ in G. Among the different tentative, the InfoGAN [165] deserved some comments. Indeed, it was implemented by reducing the information $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ when G takes only $\mathbf{z}$ and $\mathbf{c}$ as input vectors in the very first layer, but its proposition presented some issues since $\mathbf{X}_{cs}$ were too similar to $\mathbf{X}$ (any modification for $G(\mathbf{z}, \mathbf{c}_s)$ regarding to $G(\mathbf{z}, \mathbf{c})$). We observed that creating a bottleneck where $\mathbf{z}$ and $\mathbf{c}$ together make the NN to concentrate only in $\mathbf{z}$, turning $\mathbf{c}$ input completely uninformative to find $\mathbf{X}$, and so to get $\mathbf{X}_{cs}$.

Regarding the instabilities issues of training a generative network, many solutions were developed and tested. Toward this end, we observed that WGAN [77] and ALICE [155] losses show an unstable

behavior on the training for the present data set. The small number of instances and the quality of the image may block the training for these networks. The class switching task was not properly performed for any of these architectures. The $\mathbf{z}$-partitioning and the conditioned layers added solved the issue of class switching. Additionally, the SN and hinge loss greatly improve the training and the smoothness to fit the cGAAE in such an small data set. Furthermore, the WN layers borrowed from [82] provides the needed architecture expressiveness to fairly reproduce the style of each class. Finally, changing the content of a generated new image (echo shape and position) was mainly possible because of the cST, as it was shown in Chapter III by an ablation of this layer on the generative model.

Regarding the regression model developed for crack sizing and specimen parameters estimation, the aim of this study was to provide an application context to the deep neural network strategy proposed and not to perform a wide analysis to establish the best possible inverse model applied to M-TFM images thus we rely on in-house inverse deep learning model developed for M-TFM not yet published. Other inversion NN were tested for this part of the work, like [20]. We found that, up to some extent, they can profit from the enhanced data set while training. However, many existing inversion techniques may not benefit from this approach since the can not longer improve, no matter how close training data are to reality.

IV.6 . Chapter outlook and perspectives

We present a domain adaptation approach to generate realistic data from a multi-fidelity NDT&E ultrasound imaging data set. We present a new tailored cGAAE architecture for NDT&E realistic data generation proposes. The neural network is trained over a multi-label and multi-class data set for a complex weld geometry ultrasound inspection.

We demonstrate in both qualitative and quantitative ways how new generated data can be exploited for inversion tasks (i.e., regression) and enhance the prediction of neural network based inverse models. Improvements were reported in terms of prediction accuracy. This shows a practical application for the cGAAE generated data to perform a domain adaptation approach for industrial applications in NDT&E application.

The possible perspectives and axes of research interests on the cGAAE architecture proposed, are the optimization of computational burden associated to the training phase accounting for a faster spectral normalization computation like in [172]. Given the stability of the training for the realistic data generation approach presented here, adding more data, coming from experimental or simulated sources, may highly improve the accuracy of the generated images. That is, either more simulated data to treat an unbalance class representation study case, or an intermediate fidelity level coming from a FEM model may lead the cGAAE model to better convergence.
Some approaches to exploit the latent space structure are intended to be implemented in this generative model [173, 174, 175, 176, 83]. The final objective is to detach the generator to profit from the known latent space distribution to generate realistic label data. Some work in this sense seems promising [177], where an additional constrain to the latent space is added to impose a direction (Fig. IV.9) to $\mathbf{z}$ regarding a parameter p_i. Later, this direction is followed at space to generate data by changing one parameter at a time. The advantage of this approach is that only a fidelity level needs to be fully labeled during the training (e.g., the simulation data) and G can be an independent generator during the test phase. For instance, the simulated data in not need anymore, accelerating the realistic generation.

Other options may be to test architectures like in [178], where the link between the simulation parameters $\mathbf{p}$ and the $\mathcal{Z}$ space is learned by an additional NN during the adversarial training. Compared to the previous approach, this approach provides an architecture with direct access to a mapping from $\mathbf{p}$ to $\mathcal{Z}$. However, as in the previous approach, the stability of the training is not assured for the whole set of NN in the adversarial game when adding new objectives to optimize.

V - Conclusions and perspectives

V.1 . Overview

This thesis investigated the application of tailored deep learning (DL) algorithms as applied to non-destructive testing and evaluation (NDT&E) for enhancing forward and inverse problem performances with application to computationally expensive statistical studies. We proposed and tested adapted learning frameworks based on state-of-art deep learning architectures and we apply them to different NDT&E data issues representative of eddy current testing and ultrasound testing based imaging. Once deployed, the architectures are exploited as surrogate models to generate new data (i.e., generative model). These synthetic data are then used in some common inverse problems and global sensitivity analysis to show some possible applications of the proposed architectures.

More notably, the proposed approaches focus on the exploitation of data used to develop surrogate models that may account for different sources. Different simulations, experimental acquisitions, and in situ recording are examples of different possible sources. Each data source on a given NDT&E technique represents a different degree of fidelity (or similarity to the reality). More generally, the present work provides a framework to develop a surrogate model by considering the contribution of possibly many sources. This major challenge is a key part of the contribution, and it enables a complete mining of existing multi-fidelity data sets in the field of NDT&E.

Once a data set is produced for an inspection technique, the proposed methodology can be decomposed into two parts. Firstly, the development of a tailored DL architecture that learns from the initial data set is designed. Then, the architecture becomes a data generator. Secondly, the architecture shall be used to enlarge the initial dataset. Sometimes, the generation considers just one source of data (or fidelity level), and sometimes, more sources are considered to develop the architecture and later data generation.

Chapter II presents an example of a single-fidelity data set. A DL architecture based on an AutoEncoder is proposed as a generator in an eddy current testing (ECT) inspection. The initial data set is produced from a parametric simulation used as a forward solver. The data format is ECT images produced from an inspection of a specimen with two plates put together with an isolation layer. Each plate has a linear crack. Coil position, specimen proprieties, and cracks geometry parameters parametrize the simulation. A set of 12 parameters sampled from the forward solver serves as labels for a $\sim 5\,\mathrm{k}$ image data set.

Once the generating architecture is trained over the data set, the architecture replaces the forward solver to enlarge the initial data set. The initial set of parameters is the architecture input, and new labelled images can be generated from it. Thanks to the enlarged data set, the new data are used to perform extensive statistical studies such as global sensitivity analysis and feature importance ranking with high confidence levels.

Chapter III presents an example of a multi-fidelity data set. A DL architecture based on a conditional U-Net is proposed as a generator in an ultrasound testing (UT) inspection. The initial data presents a higher complexity regarding the ECT data set. The data format is also labeled images parametrized by simulation parameters. The images come from the multi-mode total focus method (M-TFM) imaging technique. The data set is a muti-fidelity set of around $5\,\mathrm{k}$ images. This time, two sources are used to produce the data : a parametric simulation as a forward solver and an experimental acquisition performed in a mock-up that reproduced the simulation parametrization. The added experimental data represent an additional complexity to developing the surrogate model.

The developed architecture for this multi-fidelity case implements similar DL techniques from the previous Chapter II to include the parameter labels as input for the data generator. Additionally, the architecture learns to translate simulation data to experience data. As in Chapter II architecture, the sur-

rogate model uses the parameters as input to generate new label data. Supplementary, the architecture infers new experimental data from the simulation. The surrogate model allows the enlargement of the initial experimental acquisitions.

Chapter IV uses the same data set from the previous Chapter. This time, a new DL architecture is proposed to tackle the problem of the stochastic nature of the experimental data. In contrast with the conditional U-Net architecture, which provides a deterministic mapping between fidelity levels on data, the generative architecture uses a statistic approach to generate an one-to-many mapping from simulation data to experimental data. This approach is better adapted to the real world since an experimental campaign generates different images from the same set of parameters since many sources of uncertainties not accounted for in simulation are present at the moment of the acquisition (e.g., probe position, environmental conditions, variability of the specimen parameters).

The stochastic data generator is used to enrich simulation data coming from a parametric forward solver. A new data set of over $15\,\mathrm{k}$ realistic images is obtained. The new data set is applied to an inverse problem for the image data set. The set of simulation parameters is retrieved from the images. The preliminary results obtained showed that the enhanced performance is shown when the newly enlarged data set is used for this task in contrast with the same inversion optimized only with data from the simulation.

V.2 . Summary of findings

V.2.1 . Tailored architectures for NDT&E small data in NDT&E

One of the challenges for NDT&E data and ML algorithms is the accessibility to data. Mainly, accessing extensive data to train the existing deep architectures in the bibliography in the NDT&E is often problematic. Generative architectures are demanding regarding samples, particularly those that use statistical approaches to learn the data distribution (called cGAAE architecture).

It is natural to think that large datasets are needed when handling increasing data complexity: a considerable number of classes, many labels, or several features in each sample. The aimed task is also linked to the need for more data. Loosely speaking, the more complex the task, the more data are required. For example, differentiating classes on a two-classes data set is relatively more straightforward when the classes are easily separated (e.g., cats versus dogs classification). The amount of data that can be representative enough for this task is less representative for a more complex aimed task (e.g., breed identification over the two classes).

In NDT&E data, the same issue exists regarding data quantity and task complexity [179]. It is worth noting that the initial data sets used through the methodology are relatively small compared to other existing data sets in the bibliography. Similar generative tasks than the proposed in the methodology exist in the bibliography. The bibliography uses large existing data sets such as MNIST, CIFAR-10, CelebA, or ImageNet. The smaller data sets in the list are MNIST, which has around $10\,\mathrm{k}$ samples for 9 classes with 28×28 features per sample; and CIFAR-10, which has $6\,\mathrm{k}$ coloured images of size 32×32 for 10 classes. Others go up to $200\,\mathrm{k}$ images with multiple labels and classes, such CelebA and ImageNet.

The used data sets in this thesis do not surpass the quantity of $6\,\mathrm{k}$ in any case. For instance, the number of classes and parameters in the UT data has 4 labels per instance: transversal wave celerity, specimen back-wall angle, and crack geometry parameters (2 labels). Moreover, as explained in Chapter III and Chapter IV, the dataset was augmented by adding a fictitious class called reconstruction mode to go from 324 samples to $\sim 5\,\mathrm{k}$ samples. The images are high-definition TFM with 128×128 features.

This gap in the number of samples between our data sets and the data sets used for DL generative approaches, in general, was one of the blocking issues to developing the proposed architectures.

V.2.2 . Tailored affine transformation for spatially correlated data on NDTE&E

A proposed solution for the problem of data accessibility in terms of quantity was the tailored architectures. The conditioned spatial transformer and the conditioned instance normalization blocks prevent the architectures from being too deep regarding the complex task : to act as a surrogate model to generate new instances. The initial tests showed that the unconditioned architectures did not get to reproduce some effects in the data (Fig. III.9) without the spatial transformer block. A possible solution to this would have been to increase the depth of the architectures, but more data was needed for this. Since the production of more data was not a realistic scenario, the problem was unblocked thanks to the tailoring of the architectures.

Adopting the Spatial Transformers (ST), FiLM and adaptative Instance Normalisation (IN) layers from the bibliography brings a solution to the lack of large data sets in the DL surrogate modelling for the NDT&E field.

The inner features representations in the Fig. II.7, III.10 and IV.9 show how those layers help the architectures to handle style and content generation, guided by the simulation parameters and the classes in the data set.

Figures III.10(c), III.10(d), IV.9(d) and IV.9(e) show how the ST in the architectures condition the latent space representation to the specimen wave celerity parameter and the back-wall angle parameter. When observing the initial data set of TFM images, one can notice that those two parameter changes represent a roto-translation in the image. This partially justifies the reason why the sub-clusters in latent representation are arranged as in the figures: two preferential directions that tend to be orthogonal per sub-cluster.

The observations showed that the more information was included in this conditioning, the better the generation quality and the faster the deep architecture training. This, together with the reached stability after including this type of layer, gives the DL frameworks the accuracy needed despite the small data available for the training.

It is important to mention that both conditional layers were conceived for the purpose of this work. The conditional spatial transformer is introduced for the first time in the bibliography in the publications produced in this thesis, and it is an adaptation of the spatial transformer used before for enhanced classification tasks in DL. Two different applications were shown: parametric conditioning and class conditioning for spatially correlated data on NDT&E.

The conditional instance normalization (FiLM and AdaIN) inner structure as applied to the application to guide a parametric conditioned generation in the Chapters II and III are also introduced, to the best of our knowledge, for the first time in the publications produced in this thesis.

V.2.3 . Different architectures proposition adapted to different data sources

The three architectures presented in the methodology were conceived considering the data set characteristic of each case. Even for frameworks with the same objective at the end (NDT&E data generation), the architectures need, up to some extent, to be adapted regarding the type of data and the generative task.

The AE backbone was the most suitable for the architecture applied to ECT single-fidelity data. Before this final architecture, an U-Net backbone was tested without success. This failed because of the skip connection in the U-Net architecture from the encoder to the decoder. This allows all the information from the input to flow directly to the neural network output. As a result, the deepest layers are not trained, and the input parameters do not condition the architecture. The same experience can be done for the cGAAE architecture when adding skip connections between the encoder and decoder. The test showed that the latent space was not informative, proving that skip connections of this nature are unsuitable for generative architectures in some generative tasks. As an alternative, ResNet connections helped the generation tasks without blocking the training of the deepest layer of the neural network.

However, the U-Net with the skip connections was successfully used for the multi-fidelity supervised case in Chapter III. Contrary to the cGAAE and the cAE, the objective of the cU-Net training is not to reconstruct the input of the neural network as in the cAE or the cGAAE, but to translate an input to a different fidelity level. In this case, the task tolerated the skip connection since the input and the output

were different.

To sum up, different architectures were adapted for different NDT&E applications and data, from a single source simulated data set to a multi-fidelity dataset. The shown application for each data set is not exclusive; for instance, the UT data set may be applied to the cAE network to accelerate the simulations, or if an ECT data set with experimental acquisition were available, the cU-Net or the cGAAE may be used to enrich the simulation data.

V.2.4 . Pseudo real-time data production for NDT&E inspections

One of the contributions derives from the fact of using ML architecture for the design of the surrogate models. Once the architectures are trained, they can be loaded into a CPU with limited memory to generate new data. This allowed the cheap generation of the realistic data set of $15\,\mathrm{k}$ images for the UT inversion task in Chapter IV and the enlarge simulated data set of $\sim 25\,\mathrm{k}$ images for the global sensitivity analysis and feature importance ranking in Chapter II.

Both generations were done once the network was trained in a short time related to the initial data set production time. For instance, an ECT forward solution takes over $40\,\mathrm{s}$ based on optimized semi-analytical models, while a DL forward solution takes $0.014\,\mathrm{s}$, when it is not parallelized. This made possible the generation of massive data needed for studies such as feature importance ranking.

V.2.5 . Application to global sensitivity analysis and feature importance ranking as NDT&E inspection study technique

SHapley Additive exPlanations (SHAP) values have gained prominence in the ML community as a method for ranking the importance of features. However, their application within the field of NDT&E is still in exploration. Surrogate model based on Deep Neural Network (DNN) architectures have not been extensively explored in NDT&E when it comes to efficiently calculating Global Sensitivity Analysis (GSA) indexes or determining Feature Importance (FI) rankings using SHAP values. In traditional machine learning research, FI methods play a crucial role in feature ranking. The emergence of SHAP values has been a significant advancement in this regard. These values, along with the "bee-swarm" plots (e.g., Fig. II.11), provide powerful tools for understanding feature importance. When applied to ECT signals, DNN regression models not only exhibit high prediction accuracy but also demonstrate exceptional computational efficiency. This efficiency is particularly noteworthy compared to the intensive computational demands of traditional forward solvers. Furthermore, adopting SHAP analysis in ECT permits a comprehensive study to be conducted in a remarkably short amount of time, allowing for deeper insights into the ranking of parameters and revealing physics-rooted behaviours. SHAP presents an interesting complementary analysis with, for instance, GSA.

V.2.6 . Relation between machine learning generative approaches and multi-fidelity data on NDT&E

During the development of each architecture, some concepts commonly used in ML in the computer vision research community were applied to the NDT&E data. Style and content concepts used in ML as been translated to to the equivalent notions of noise (as intended as a quantity that cannot be directly simulated via numerical solvers) and flaw signature (as intended as the quantity that impacts more the outcomes of the numerical simulations) in ECT and TFM images. This link has been exploited to propose a new approach towards high-quality data generation on NDT&E.

The observed differences features between different fidelity data on NDT&E (e.g., simulation and experimental) were compared to the differences recalled in the ML when translating images from different domains [79] and integrating them into our NDT&E data generation approach fully.

V.2.7 . Thesis work contributions accordingly to the research directions in the scientific ML framework

Scientific machine learning [180] is a research topic dealing with the integration of numerical simulation, scientific knowledge and machine learning into a common framework aiming at enhancing the outcomes of the results and the analysis of each of the mentioned research directions as taken separately.

This thesis addresses some of the priority research directions in the scientific ML fields as applied to NDT&E. In the technical report from [180], the authors mention that incorporating scientific domain knowledge has the potential to reduce data requirements dramatically. An example of this is how we tackle the lack of large data sets by proposing tailored architectures where the inner layers are adapted to the NDT&E domain.

Robust and interpretable scientific machine learning is also a recurrent subject. In the methodology proposed in this thesis, explainability has been boarded through the inner features exportation and interposed with the domain knowledge to shed some light on how surrogate models work in a physic-rooted data set. Together with this exploration, the validation of each surrogate model gives an idea of the robustness of the methodology applied in this thesis.

The term "outer loop" is used increasingly to describe computational applications that form outer loops around a forward simulation. Those applications often rely on machine learning-enhance modelling and intelligent automation. Examples of this have been treated in specific NDT&E techniques in this thesis. The surrogate models have been applied in some diagnostic and evaluation techniques to prove the applicability of machine learning-enhance modelling in the NDT&E domain. There is a possibility to extrapolate this methodology to other fields where the same need exists.

V.3 . Future work

V.3.1 . Application to NDT&E and SHM techniques other than ECT or UT

The frameworks are not exclusive to the NDT&E methods presented in the methodology, and this means that they can be extended to other NDT&E data more than ECT and UT techniques. For instance, Infrared Thermography, Acoustic Emission Testing, and Structural Health Monitoring may be a source of input data to be deployed in the proposed architectures. The objective may be the same: fast and controlled realistic data generation. Imaging techniques from those may be more adapted for the architectures, essentially because the spatial transformer block focuses on convolution neural networks for images.

V.3.2 . Application to NDT&E and SHM data other than images

The tailored architectures are conceived for image data. However, it is possible to find several similar layers and blocks in the DL bibliography to conceive new tailored architectures more adapted to, for instance, temporal signals (often found in NDT&E data). The key regarding the tailored architectures is to provide the neural network a means to link the feature to a dedicated transformation. For instance, the spatial transformer is a set of affine transformations from a set of rotation, translation, dilation, and shear transformations that links the extracted features to the simulation parameter in the cAE or the cU-Net. For a time-dependent signal, some layers as [181][182][183] based on frequency domain transformations may be adapted in the same sense to link temporal signals and parameters.

V.3.3 . Physics informed spatial transformers and conditioned instance normalization

A promising intuition and contribution of this work is the possibility of adding the concept of physics-informed learning to the architectures. The latent space representation in all architectures showed that the ST and the conditioned instance normalization learned the physic relations of the parameters and the image features. For instance, the spatial transformer allowed the cU-Net (See Appendix VIII) to shift in a coherent way when a change in the celerity is set in the input. Similarly, the cAE performs the rotation in the flaw signature in a coherent way when changing the angle between cracks (Fig. II.6). In [184], they describe three ways to introduce physics in ML surrogate models as applied to imaging. In our case, the ML architecture conception is guided by the physics (last column in Fig. V.1) combined with a dedicated reconstruction loss (middle column in Fig. V.1).

However, a possibility to accelerate training time and to ensure the optimization stability is to introduce physic-guided loss either as an output error metric or an inner layer metric. The conditioned ST and the conditioned IN are the links between the input parameters (called measurement in the Fig. V.1),

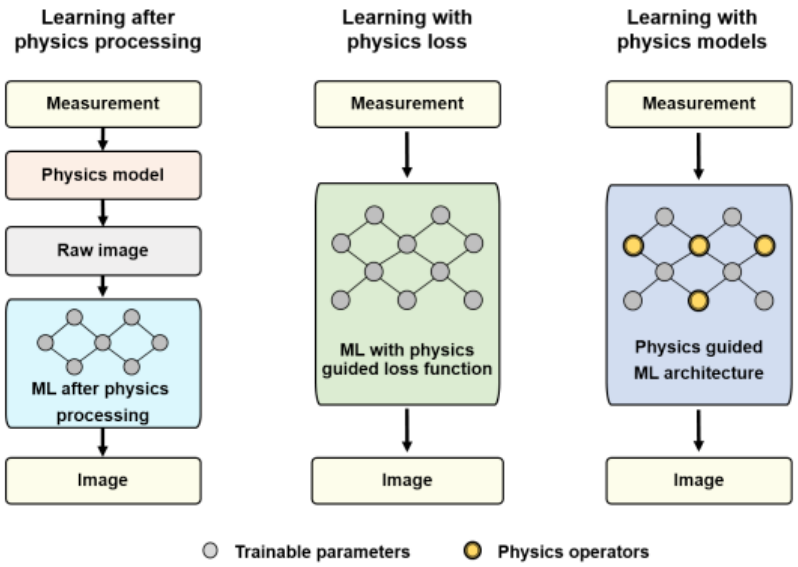


Figure V.1. Three ways of incorporating physics into the ML model. (a) Learning after physics processing: a physic model is employed to initialize the input of ML models. (b) Learning with physics loss: physics knowledge is incorporated into the loss functions. (c) Learning with physics models: physics knowledge is used to guide the design of ML architecture. Image borrowed from [184].

and the resulting generated image. It is possible to think to constrain the learning of the weights of the inner dense layer in both ST and IN. The constrain should be set in terms of physics-rooted criteria.

For instance, the cU-Net showed poor performance when interpolating out of the learned range of parameters. A suitable constraint to generate not only on the parameter range but also to extrapolate would be to keep coherent energy (measured in the image) regarding the changes in the specimen celerity T as a parameter. If a metric can be conceived for this, this would improve the extrapolation for the surrogate model for TFM images.

Regarding the architecture optimization, a possibility to accelerate the training is to explore the localization network of the ST layers once the architecture is trained. For instance, the activation values of the dense networks may stay invariant for a parameter. This exploration may indicate two interesting things: first, the number of ST layers is too much for the application; secondly, the parameter does not need any affine transformation to represent its impact in the output image. The last point can be a departing point to better contribute to the explainability of the tailored architectures and better understand how a parameter plays a role in the inspection.

V.3.4 . cGAAE stochastic generator as a surrogate model by latent space exploration

During the test on the cGAAE, some tests to constrain the latent space (Fig. IV.9) structure were undertaken. The objective was to find a way to decouple the encode and decoder to generate TFM images only using the encoder. This approach would enable to dispense with the simulation image input to generate realistic samples. The test was done over one of the parameters (celerity T) using the approach proposed by [174, 175, 83]. The preliminary results show that it is possible to force one direction over the latent space. This imposed direction is correlated to the parameter. In other words, the decoder 'knows' how to generate images when the chosen parameter varies.

In Fig. V.2 an example of generation direction imposed in the latent space. The loss used a τ unitary vector to fix a direction regarding the parameter c_T. Only the labels from the simulation are used for this purpose. x_s and x'_s are a couple of simulation images with the same labels except the c_T label. For the couples, c_T and c'_T values are as close as possible regarding the available data in the training set.

Instead of imposing a structure by the loss, the latent space in Fig. IV.9 can be explored after training

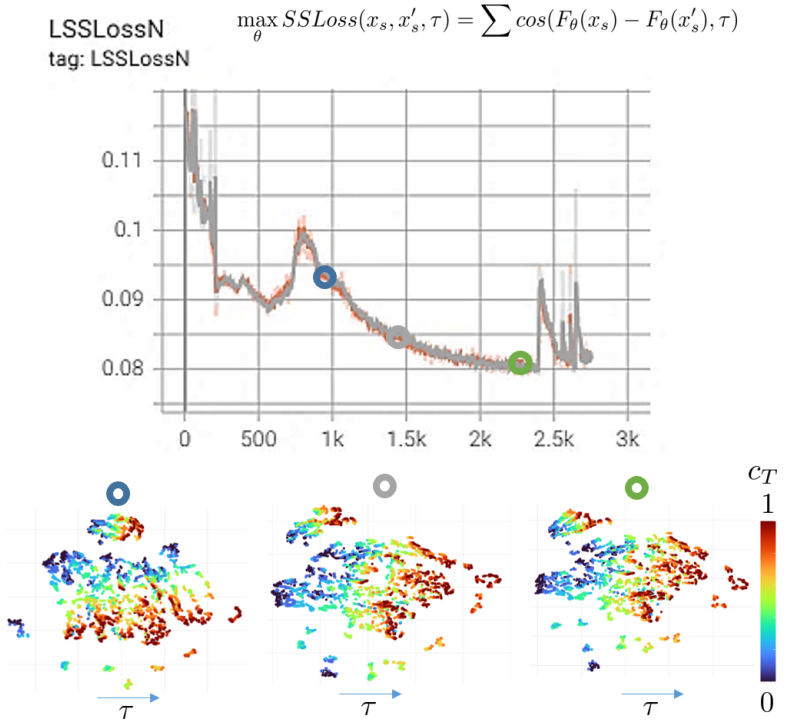


Figure V.2. $\mathcal{Z}$ latent space exploration. Imposed direction pour parameter c_T during the training with the Self-supervised Longitudinal Loss proposed by [177]. τ is a direction in the $\mathcal{Z}$ space imposed by the loss exposed in the figure. The convergence of the loss is shown for the adversarial training. Above, different stages of $\mathcal{Z}$ structure during the training are shown. The coloured circle marks correspond to different epochs in the loss curve. The celerity values of the latent space are normalized between 0 and 1.

by techniques like Partial Least Squares (PLS) or supervised Principal Component Analysis (PCA) [185]. The idea is to find a correlation between $\mathcal{Z}$ and the parametric space $\mathcal{P}$.

V.3.5 . Other possible applications

This implicit characteristic for DL algorithms mentioned on Subsection V.2.4 makes possible the application of the trained models in, for instance, on NDT&E data real-time simulation. The exposed latency for generation from the surrogate models promises a suitable solution for quasi real-time applications.

An example is where the user interacts with virtual simulators. Some platforms designed for education and training on NDT&E can largely profit from these approaches. The common solution in these platforms is to produce simulated data and to stock it in a data base. The data are later read to be displayed to the user. This solution denotes two important limitations: data are static, and not all representative samples that the user shall demand can be stocked in advance, so the user experience is limited to the data base produced before. Secondly, a closely related issue is the portability of the data base regarding to the storage space. The presented architecture represents a solution for the enumerated problem. The DL neural networks can be run in a CPU with reduced storage and memory to generate NDT&E data in real-time and on demand.

Similarly, a possible application may be the NDT&E simulated data base acceleration. The frameworks can accelerate tools like CIVA or FEM solvers if they learn from an initial reduced data set produced by the solver to generate more data later. For this purpose, it may be interesting to study the sampling of the parameter strategies space and quantification of instances required to converge in the DL architecture for a broad set of inspection techniques.

V.3.6 . Many fidelity data sources for cGAAE

The cGAAE architecture was conceived to contain several sources of data. Relatively simple and straightforward adaptation may be made to the architecture of the cGAAE to introduce more fidelity levels: adding a class label may be enough by training over the same losses and network. This may be useful if intermediate fidelity data are added to the multi-fidelity UT data set. In this case, the projection discriminator embedding and the number of filters treated by the ST in the generator may increase to make a place for new unseen features from multiple fidelity levels.

V.3.7 . Toward the diffusion models application on NDT&E and SHM

A less explored perspective for the developed architecture is the application of diffusion models [186], particularly the stable diffusion model. The advantage of these models is that they outperform the generation quality of GANs or similar architectures. As an example, DALL-E 2 [187] uses stable diffusion models. Stable diffusion models are a type of diffusion model that is more stable and efficient to train than other types of diffusion models.

Stable diffusion models work by gradually removing noise from a latent representation of the data. This is done by iteratively applying a denoising function to the latent representation. The denoising function is trained to remove noise from the latent representation while preserving the important features of the data. The interesting characteristic of the stable diffusion model is the ability to generate realistic images from a wide variety of text descriptions. In the case of NDT&E data, instead of text, a dictionary of parameters can be the input to generate high-quality images for an inspection procedure.

A draft of how the application may look is in the Fig. V.3. Stable diffusion models used a set of transformers for text input in an U-Net architecture. The architecture takes a noise latent representation map and iterates it to generate a conditioned latent representation. The U-Net generates an informative latent vector after iteration. The latent vector is then used as generator input that creates a high-quality image. In the case of a NDT&E application, text can be replaced by a set of parameters or a class embedding. The ST and FiLM then replace the advanced transformers for text-to-image ML algorithms in our cU-Net architecture.

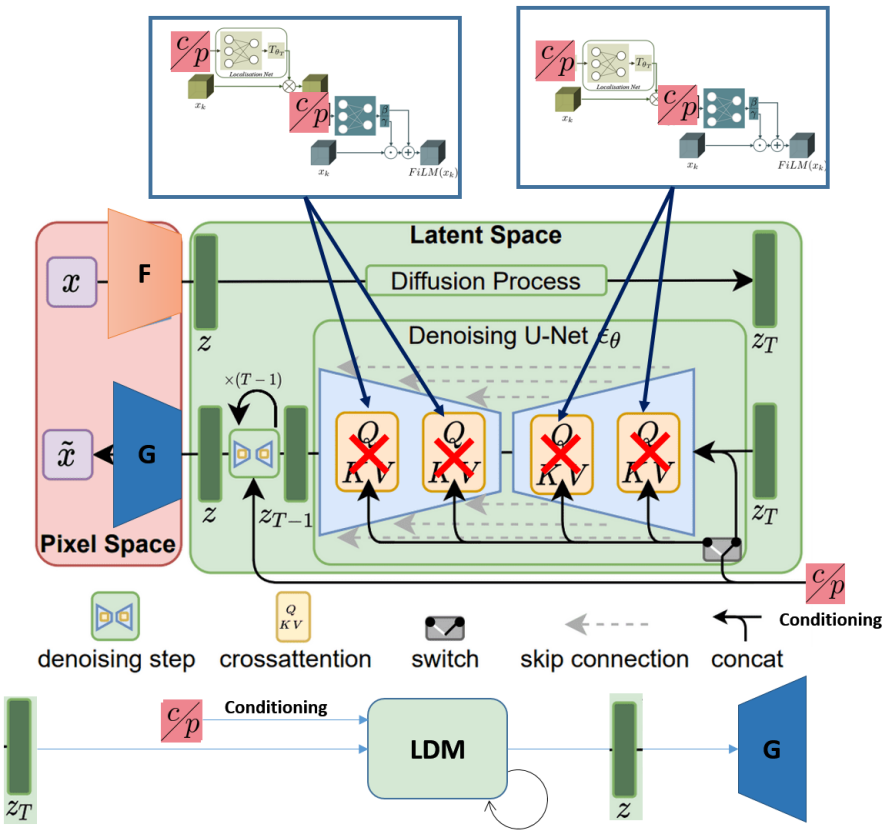


Figure V.3. Draft for stable diffusion model c-Unet application. Figure modified from [186].

Bibliography

[1] Hongbin Sun, Pradeep Ramuhalli, and Richard Jacob. Machine learning for ultrasonic nondestructive examination of welding defects: A systematic review. *Ultrasonics*, page 106854, 2022. 0, III.1

[2] Jiaxing Ye, Shunya Ito, and Nobuyuki Toyama. Computerized ultrasonic imaging inspection: From shallow to deep learning. *Sensors*, 18(11):3820, 2018. 0, III.1

[3] Nauman Munir, Hak-Joon Kim, Jinhyun Park, Sung-Jin Song, and Sung-Sik Kang. Convolutional neural network for ultrasonic weldment flaw classification in noisy conditions. *Ultrasonics*, 94:74–81, 2019. 0, III.1

[4] Richard J. Pyle, Rhodri L.T. Bevan, Robert R. Hughes, Rosen K. Rachev, Amine Ait Si Ali, and Paul D. Wilcox. Deep learning for ultrasonic crack characterization in nde. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 3010(c), 2020. 0, I.6.1, III.1

[5] Luca Rosafalco, Andrea Manzoni, Stefano Mariani, and Alberto Corigliano. Fully convolutional networks for structural health monitoring through multivariate time series classification. *Adv. Model. Simul. Eng. Sci.*, 7(1):1–31, 2020. 0, III.1

[6] Roberto Miorelli, Clement Fisher, Andrii Kulakovskyi, Bastien Chapuis, Olivier Mesnil, and Oscar D'Almeida. Defect sizing in guided wave imaging structural health monitoring using convolutional neural networks. *NDT & E Int.*, page 102480, 2021. 0, III.1

[7] Roberto Miorelli, Andrii Kulakovskyi, Bastien Chapuis, Oscar D'Almeida, and Olivier Mesnil. Supervised learning strategy for classification and regression tasks applied to aeronautical structural health monitoring problems. *Ultrasonics*, 113:106372, 05 2021. 0, III.1

[8] Long Bai, Florian Le Bourdais, Roberto Miorelli, Pierre Calmon, Alexander Velichko, and Bruce W Drinkwater. Ultrasonic defect characterisation using the scattering matrix: A performance comparison study of bayesian inversion and machine learning schemas. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 2021. 0, III.1

[9] Sergio Cantero-Chinchilla, Paul D Wilcox, and Anthony J Croxford. Deep learning in automated ultrasonic nde–developments, axioms and opportunities. *NDT & E Int.*, page 102703, 2022. 0, III.1, IV.1

[10] Joel B. Harley and Daniel Sparkman. Machine learning and NDE: Past, present, and future. *AIP Conf. Proc.*, 2102(1), 05 2019. 090001. 0, IV.1

[11] Agnimitra Sengupta, Hoda Azari, Ilgin Guler, and Parisa Shokouhi. Transfer learning of Impact Echo signal classification from laboratory to the field. *e-J. Nondestr. Test.*, 27(9):16–19, 2022. 0, I.7.7, IV.1

[12] I. Dewa Made Oka Dharmawan, Jinyi Lee, and Sunbo Sim. Defect Shape Classification Using Transfer Learning in Deep Convolutional Neural Network on Magneto-Optical Nondestructive Inspection. *Appl. Sci.*, 12(15), 2022. 0, I.7.7, IV.1

[13] Xiaopeng Wang and Xinghua Yu. Understanding the effect of transfer learning on the automatic welding defect detection. *NDT & E Int.*, 134(December 2022):102784, 2023. 0, I.7.7, IV.1

[14] P. Gardner, R. Fuentes, N. Dervilis, C. Mineo, S. G. Pierce, E. J. Cross, and K. Worden. Machine learning at the interface of structural health monitoring and non-destructive evaluation: Machine

Learning in SHM and NDE. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2182), 2020. 0, I.7.8, IV.1

[15] Mahta HassanPour Zonoozi and Vahid Seydi. A Survey on Adversarial Domain Adaptation. *Neural Processing Letters*, 55(3):2429–2469, 2022. 0, IV.1

[16] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis Bach and David Blei, editors, *Proc. 32nd Int. Conf. Mach. Learn.*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France, 07–09 Jul 2015. PMLR. 0, I.7.8, IV.1

[17] Thulsiram Gantala and Krishnan Balasubramaniam. Automated Defect Recognition for Welds Using Simulation Assisted TFM Imaging with Artificial Intelligence. *J. Nondestruct. Eval.*, 40:28, 2021. 0, I.7.8, III.1, IV.1

[18] Luka Posilovic, Duje Medak, Marko Subasic, Marko Budimir, and Sven Loncaric. Generative adversarial network with object detector discriminator for enhanced defect detection on ultrasonic b-scans. *Neurocomputing*, 459:361–369, 2021. 0, I.7.8, III.1, IV.1

[19] Liangliang Cheng and Mathias Kersemans. Dual-irt-gan: A defect-aware deep adversarial network to perform super-resolution tasks in infrared thermographic inspection. *Compos. Part B Eng.*, 247:110309, 2022. 0, I.7.8, IV.1

[20] Richard J. Pyle, Rhodri L.T. Bevan, Robert R. Hughes, Amine Ait Si Ali, and Paul D. Wilcox. Domain Adapted Deep-Learning for Improved Ultrasonic Crack Characterization Using Limited Experimental Data. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 69(4):1485–1496, 2022. 0, I.7.8, III.1, IV.1, IV.5

[21] P. Calmon, S. Mahaut, S. Chatillon, and R. Raillon. CIVA: An expertise platform for simulation and processing NDT data. *Ultrasonics*, 44(SUPPL.):e975–e979, 12 2006. 0, I.4

[22] V. Dorval, F. Jenson, G. Corneloup, and J. Moysan. Accounting for structural noise and attenuation in the modeling of the ultrasonic testing of polycrystalline materials. *AIP Conf. Proc.*, 1211:1309–1316, 2010. I.2

[23] X. Bai, B. Tie, J. H. Schmitt, and D. Aubry. Comparison of ultrasonic attenuation within two- and three-dimensional polycrystalline media. *Ultrasonics*, 100(June), 2020. I.2

[24] Long Bai, Florian Le Bourdais, Roberto Miorelli, Pierre Calmon, Alexander Velichko, and Bruce W. Drinkwater. Ultrasonic defect characterisation using the scattering matrix: A performance comparison study of Bayesian inversion and machine learning schemas. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 2021. I.2

[25] Roberto Miorelli and Christophe Reboud. Adaptive sparse grid approach for the efficient simulation of pulsed eddy current testing inspections. In *AIP Conference Proceedings*, volume 1949. American Institute of Physics Inc., 04 2018. I.2

[26] Joscha Maier, Stefan Sawall, Michael Knaup, and Marc Kachelrieß. Deep Scatter Estimation (DSE): Accurate Real-Time Scatter Estimation for X-Ray CT Using a Deep Convolutional Neural Network. *J. Nondestruct. Eval.*, 37(3):1–9, 2018. I.2

[27] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 02 2019. I.2

[28] Javier García-Martín, Jaime Gómez-Gil, and Ernesto Vázquez-Sánchez. Non-destructive techniques based on eddy current testing. *Sensors*, 11(3):2525–2565, 2011. I.3

[29] C Gilles-Pascaud, G Pichenot, D Premel, C Reboud, and A Skarlatos Cea. Modelling of Eddy Current inspections with CIVA. pages 25–28, 2008. I.3.1

[30] Roberto Miorelli, Christophe Reboud, Theodoros Theodoulidis, John Martinos, Nikolaos Poulakis, and Dominique Lesselier. Coupled approach vim–bem for efficient modeling of ect signal due to narrow cracks and volumetric flaws in planar layered media. *NDT&E Int.*, 62:178–183, 2014. I.3.1

[31] Anastassios Skarlatos. A mixed spatial–spectral eddy-current formulation for pieces with one symmetry axis. *IEEE Transactions on Magnetics*, 56(9):1–11, 2020. I.3.1

[32] Xin-Qing Sheng and Wei Song. *Method of Moments*, pages 29–151. 2012. I.3.1

[33] Theodoros Theodoulidis. Developments in efficiently modelling eddy current testing of narrow cracks. *NDT and E International*, 43(7):591–598, 2010. I.3.1

[34] Roberto Miorelli, Christophe Reboud, Dominique Lesselier, and Theodoros Theodoulidis. Eddy current modeling of narrow cracks in planar-layered metal structures. *IEEE Transactions on Magnetics*, 48(10):2551–2559, 2012. I.3.1, II.2.1, II.4.1

[35] Konstantinos Pipis, Anastassios Skarlatos, Theodoros Theodoulidis, and Dominique Lesselier. Ect-signal calculation of cracks near fastener holes using an integral equation formalism with dedicated green's kernel. *IEEE Transactions on Magnetics*, 52(4):1–8, 2016. I.3.1

[36] Nicolas Gengembre, Alain Lhémery, Ryuji Omote, Thierry Fouquet, and Andreas Schumm. A semi-analytic-fem hybrid model for simulating ut configurations involving complicated interactions of waves with defects. *AIP Conference Proceedings*, 700(1):74–80, 02 2004. I.4

[37] Alexandre Imperiale, Sylvain Chatillon, Pierre Calmon, Nicolas Leymarie, Sébastien Imperiale, and Edouard Demaldent. Ut simulation of embedded parametric defects using a hybrid model based upon spectral finite element and domain decomposition methods. 2016. I.4.1

[38] Alexandre Imperiale, Sylvain Chatillon, Michel Darmon, Nicolas Leymarie, and Edouard Demaldent. UT simulation using a fully automated 3D hybrid model: Application to planar backwall breaking defects inspection. *AIP Conference Proceedings*, 1949(1):050004, 04 2018. I.4.1, I.6

[39] Caroline Holmes, Bruce W. Drinkwater, and Paul D. Wilcox. Post-processing of the full matrix of ultrasonic transmit–receive array data for non-destructive evaluation. *NDT & E Int.*, 38(8):701–711, December 2005. I.4.1, III.1

[40] Mark Sutcliffe, Miles Weston, Ben Dutton, Peter Charlton, and Kelvin Donne. Real-time full matrix capture for ultrasonic non-destructive testing with acceleration of post-processing through graphic hardware. *NDT and E International*, 51:16–23, 10 2012. I.4.1

[41] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018. I.5

[42] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. I.6

[43] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*. 2019. I.1, I.10, I.13

[44] Luka Posilovic, Duje Medak, Marko Subasic, Tomislav Petkovic, Marko Budimir, and Sven Loncaric. Flaw detection from ultrasonic images using YOLO and SSD. In *International Symposium on Image and Signal Processing and Analysis, ISPA*, volume 2019-Septe, pages 163–168. IEEE Computer Society, sep 2019. I.6.1, I.7.4

[45] Roberto Miorelli, Clement Fisher, Andrii Kulakovskyi, Bastien Chapuis, Olivier Mesnil, and Oscar D'Almeida. Defect sizing in guided wave imaging structural health monitoring using convolutional neural networks. *NDT & E Int.*, page 102480, 2021. I.6.1, I.7.4, III.1

[46] M. G.R. Sause, A. Gribov, A. R. Unwin, and S. Horn. Pattern recognition approach to identify natural clusters of acoustic emission signals. *Pattern Recogn. Lett.*, 33(1):17–23, jan 2012. I.6.2

[47] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philos. Mag. J. Sci.*, 2(11):559–572, 1901. I.6.2, VI.4

[48] Pierre Comon. Independent Component Analysis. In J-L.Lacoume, editor, *Higher-Order Statistics*, pages 29–38. Elsevier, June 1992. I.6.2

[49] Michael Ghil, M.R. Allen, Michael Dettinger, Kayo Ide, D. Kondrashov, Michael Mann, Andrew Robertson, A. Saunders, Yudong Tian, Ferenc Varadi, and P. Yiou. Advanced spectral methods for climate time series. *Rev. Geophys.*, 2002:1003–1043, 01 2002. I.6.2

[50] Manabu Kotani, Hiroki Takabatake, and Seiichi Ozawa. Supervised independent component analysis with class information. In Nikhil Ranjan Pal, Nik Kasabov, Rajani K. Mudi, Srimanta Pal, and Swapan Kumar Parui, editors, *Neural Information Processing*, pages 1052–1057, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. I.6.2

[51] Raymond Kurzweil. The age of intelligent machines. In *MIT Press - Current Science*, number 64(6), pages 434–436. Juhu, Bombay, 1990. I.7

[52] P.H. Winston. *Artificial Intelligence*, volume 3rd ed. 1992. I.7

[53] Pariwat Ongsulee. Artificial intelligence, machine learning and deep learning. In *2017 15th international conference on ICT and knowledge engineering (ICT&KE)*, pages 1–6. IEEE, 2017. I.12

[54] Julius Berner, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. The Modern Mathematics of Deep Learning. *Mathematical Aspects of Deep Learning*, pages 1–111, 2022. I.7

[55] Nikhil Buduma and Nicholas Locascio. *Fundamentals of deep learning : Designing Next-Generation Machine Intelligence Algorithms*. 2017. I.13

[56] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021. I.13

[57] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, pages 448–456, 2015. I.7.3

[58] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. I.7.3

[59] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. I.7.3

[60] Xinghui Dong, Christopher J. Taylor, and Tim F. Cootes. Defect Detection and Classification by Training a Generic Convolutional Neural Network Encoder. *IEEE Trans. Signal Process.*, 68:6055–6069, 2020. I.7.4

[61] Mostafa Elsaadouny, Jan Barowski, and Ilona Rolfes. A convolutional neural network for the non-destructive testing of 3D-printed samples. In *International Conference on Infrared, Millimeter, and Terahertz Waves, IRMMW-THz*, volume 2019-September. IEEE Computer Society, 09 2019. I.7.4

[62] Kushal Virupakshappa and Erdal Oruklu. Multi-Class Classification of Defect Types in Ultrasonic NDT Signals with Convolutional Neural Networks. In *IEEE International Ultrasonics Symposium, IUS*, volume 2019-October, pages 1647–1650. IEEE Computer Society, 10 2019. I.7.4

[63] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. I.7.4

[64] S. Ben Driss, M. Soua, R. Kachouri, and M. Akil. A comparison study between MLP and convolutional neural network models for character recognition. *Real-Time Image and Video Processing 2017*, 10223:1022306, 2017. I.7.4

[65] Edgar Medina, Mariane R. Petraglia, Jose Gabriel R.C. Gomes, and Antonio Petraglia. Comparison of CNN and MLP classifiers for algae detection in underwater pipelines. *Proceedings of the 7th International Conference on Image Processing Theory, Tools and Applications, IPTA 2017*, 2018-January:1–6, mar 2018. I.7.4

[66] Sander Greenland, Mohammad Ali Mansournia, and Douglas G Altman. Sparse data bias: a problem hiding in plain sight. *BMJ*, 352, 2016. I.7.5

[67] Sander Greenland, Judith A Schwartzbaum, and William D Finkle. Problems due to Small Samples and Sparse Data in Conditional Logistic Regression Analysis A MATCHED-PAIR STUDY OF CHILDHOOD CANCER. 151(5), 2000. I.7.5

[68] L J P Van Der Maaten, E O Postma, and H J Van Den Herik. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10:1–41, 2009. I.7.5

[69] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering T-SNE, UMAP, TriMap, and PaCMAP for data visualization. *Journal of Machine Learning Research*, 22:1–73, 2021. I.7.5

[70] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *J. Mach. Learn. Res.*, 9:2579–2605, 2008. I.7.5.1, II.4.2, III.3.3, IV.4.1

[71] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. I.7.5.1

[72] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. 2020. I.7.5.1

[73] M. A. Kramer. Autoassociative neural networks. *Computers & Chemical Engineering*, 16(4):313–328, apr 1992. I.7.5.2

[74] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014. I.7.5.2

[75] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 3, pages 2672–2680, 2014. I.7.6

[76] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv e-prints*, (Mmd):1–17, 2017. I.7.6, IV.3

[77] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 1, pages 298–321, 2017. I.7.6, IV.5

[78] Leon Gatys, Alexander Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. *J. Vis.*, 2016. I.7.6

[79] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural Style Transfer: A Review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385, nov 2020. I.7.6, V.2.6

[80] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms, 2017. I.7.6

[81] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls, 2020. I.7.6

[82] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 4396–4405, 2019. I.7.6, IV.1, IV.3.1, IV.3.1.1, IV.3.1.1, IV.3.1.2, IV.5

[83] Xiao Li, Chenghua Lin, Ruizhe Li, Chaozheng Wang, and Frank Guerin. Latent Space Factorisation and Manipulation via Matrix Subspace Projection. Technical report, 2020. I.7.6, IV.6, V.3.4

[84] Romit Maulik, Arvind Mohan, Bethany Lusch, Sandeep Madireddy, Prasanna Balaprakash, and Daniel Livescu. Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena*, 405:132368, 2020. I.7.6

[85] William Peebles, John Peebles, Jun-Yan Zhu, Alexei Efros, and Antonio Torralba. The Hessian Penalty: A Weak Prior for Unsupervised Disentanglement. (August):581–597, 2020. I.7.6

[86] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016. I.7.6

[87] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Med. Image Comput. and Comput.*, pages 234–241, Cham, 2015. Springer Int. Publishing. I.7.6

[88] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. I.7.6

[89] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2023. I.7.6

[90] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proc. of the IEEE*, 109(1):43–76, 2021. I.7.7

[91] Weihua Li, Ruyi Huang, Jipu Li, Yixiao Liao, Zhuyun Chen, Guolin He, Ruqiang Yan, and Konstantinos Gryllias. A perspective survey on deep transfer learning for fault diagnosis in industrial scenarios: Theories, applications and challenges. *Mech. Syst. Signal Process.*, 167:108487, 2022. I.7.8, IV.1

[92] Gerardo E. Granados, Roberto Miorelli, Filippo Gatti, and Didier Clouteau. A deep learning framework for efficient global sensitivity analysis and shap values calculations applied to eddy current testing problems. 2023. Accepted for publication. I.8

[93] G.E. Granados, R. Miorelli, F. Gatti, S. Robert, and D. Clouteau. Towards a multi-fidelity deep learning framework for a fast and realistic generation of ultrasonic multi-modal total focusing method images in complex geometries. *NDT & E International*, page 102906, 2023. I.8

[94] G.E. Granados, F. Gatti, R. Miorelli, S. Robert, and D. Clouteau. Generative domain-adapted adversarial auto-encoder model for enhanced ultrasonic imaging applications. *Engineering Applications of Artificial Intelligence*, 2023. Submitted. I.8

[95] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B Stephenson, and T. Wagener. Sensitivity analysis of environmental models: A systematic review with practical workflow. *Environ. Modell. Software*, 79:214–232, 2016. II.1, II.3

[96] J. Nagawkar and L. Leifsson. Efficient global sensitivity analysis of model-based ultrasonic nondestructive testing systems using machine learning and sobol'indices. *J. Nondestr. Eval. Diagn. Progn. Eng. Syst.*, 4(4), 2021. II.1, II.2

[97] J. C. Aldrin, E. K. Oneida, E. B. Shell, H. A. Sabbagh, E. Sabbagh, R. K. Murphy, S. Mazdiyasni, E. A. Lindgren, and R. D. Mooers. Model-based probe state estimation and crack inverse methods addressing eddy current probe variability. In *AIP Conf. Proc.*, volume 1806, page 110013, 2017. II.1

[98] J. C Aldrin, E. A Lindgren, and D. S. Forsyth. Intell. augmentation in nondestr. eval. In *AIP Conf. Proc.*, volume 2102, page 020028, 2019. II.1

[99] Xiaosong Du and Leifur Leifsson. Multifidelity model-assisted probability of detection via cokriging. *NDT & E Int.*, 108:102156, 2019. II.1, II.2, III.2

[100] Caifang Cai, Roberto Miorelli, Marc Lambert, Thomas Rodet, Dominique Lesselier, and Pierre-Emile Lhuillier. Metamodel-based markov-chain-monte-carlo parameter inversion applied in eddy current flaw characterization. *NDT & E Int.*, 99:13–22, 2018. II.1, III.5

[101] S. Bilicz, M. Lambert, and Sz Gyimothy. Kriging-based generation of optimal databases as forward and inverse surrogate models. *Inverse Probl.*, 26(7):074012, 2010. II.1

[102] S. Ahmed, R. Miorelli, M. Salucci, and A. Massa. Real-time flaw characterization through learning-by-examples techniques: A comparative study applied to ect. In *Electromagn. Nondestr. Eval.*, pages 228–235. IOS Press, 2017. II.1

[103] R. Miorelli, X. Artusi, and C. Reboud. An efficient adaptive database sampling strategy with applications to eddy current signals. *Simulation Modelling Practice and Theory*, 80:75–88, 2018. II.1

[104] Roberto Miorelli, Xavier Artusi, Anis Ben Abdessalem, and Christophe Reboud. Database generation and exploitation for efficient and intensive simulation studies. *AIP Conference Proceedings*, 1706(1):180002, 02 2016. II.1

[105] J. Nagawkar and L. Leifsson. Applications of polynomial chaos-based cokriging to simulation-based analysis and design under uncertainty. In *Int. Des. Eng. Tech. Conf. and Comput. and Inf. in Eng. Conference*, 2020. II.1

[106] X. Du and L. Leifsson. Multifidelity modeling by polynomial chaos-based cokriging to enable efficient model-based reliability analysis of ndt systems. *J. Nondestr. Eval.*, 39(1):13, 2020. II.1, II.2

[107] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. 2016. II.1, II.2.1

[108] A. Bingler and S. Bilicz. Sensitivity analysis using a sparse grid surrogate model in electromagnetic nde. In *22nd International Workshop on Electromagnetic Nondestructive Evaluation (ENDE 2017)*, France, September 6–8 2019. II.1

[109] Extende. Civa platform. https://www.extende.com/civa-in-a-few-words, 2023. Accessed: April 2023. II.1, II.4.1

[110] S. Lundberg, S. M .and Lee. A unified approach to interpreting model predictions. *Adv. in neural Inf. Process. Syst.*, 30, 2017. II.1, II.3, II.5

[111] Xiaosong Du and Leifur Leifsson. Multifidelity model-assisted probability of detection via cokriging. *NDT & E Int.*, 108:102156, 2019. II.2

[112] X. Du, L. Leifsson, W. Meeker, P. Gurrala, J. Song, and R. Roberts. Efficient model-assisted probability of detection and sensitivity analysis for ultrasonic testing simulations using stochastic metamodeling. *J. of Nondestr. Eval., Diagnog. and Progn. of Eng. Syst.*, 2(4), 2019. II.2, III.5

[113] V. Nerlikar, O. Mesnil, R. Miorelli, and O. D'almeida. Damage detection with ultrasonic guided waves using machine learning and aggregated baselines. *Struct. Health Monit.*, April 2023. II.2

[114] R. Miorelli, C. Reboud, T. Theodoulidis, N. Poulakis, and D. Lesselier. Efficient modeling of ECT signals for realistic cracks in layered half-space. *IEEE Trans. Magn.*, 49(6):2886–2892, June 2012. II.2.1, II.4.1

[115] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proc. IEEE Int. Conf. Comput. Vis.*, 2015 Inter:1026–1034, 2015. II.2.1, III.2.2

[116] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI Conf. Proc. on Artif. Intell.*, volume 32, 2018. II.2.1, III.2

[117] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *Adv. in Neural Inf. Proc. Syst.*, 2015-Janua:2017–2025, 2015. II.2.1, III.2, III.2.2, IV.3.1.1

[118] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. II.2.2, III.2.2, IV.1, IV.3.1.1

[119] Haris Iqbal. Harisiqbal88/plotneuralnet v1.0.0, December 2018. II.2

[120] Ilya M. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Math. Comput. Simul*, 55(1-3):271–280, 2001. II.3

[121] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008. II.3

[122] E. Borgonovo et al. A new uncertainty importance measure. *Reliab. Eng. Syst. Saf.*, 92(6):771–784, 2007. II.3

[123] D. P. Kingma and Jimmy B. Adam: A method for stochastic optimization, 2017. II.4.1

[124] *Principal Component Analysis for Special Types of Data*, pages 338–372. Springer New York, New York, NY, 2002. II.4.2

[125] T. Iwanaga, W. Usher, and J. Herman. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. *Socio-Environ. Sys. Modell.*, 4:18155, 2022. II.5

[126] Lucas Merabet, Sébastien Robert, and Claire Prada. The multi-mode plane wave imaging in the Fourier domain: Theory and applications to fast ultrasound imaging of cracks. *NDT & E Int.*, 110:102171, March 2020. III.1, III.2.1

[127] Léonard Le Jeune, Sébastien Robert, Eduardo Lopez Villaverde, and Claire Prada. Plane Wave Imaging for ultrasonic non-destructive testing: Generalization to multimodal imaging. *Ultrasonics*, 64:128–138, January 2016. III.1

[128] Rhodri L. T. Bevan, Nicolas Budyn, Jie Zhang, Anthony J. Croxford, So Kitazawa, and Paul D. Wilcox. Data fusion of multiview ultrasonic imaging for characterization of large defects. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 67(11):2387–2401, 2020. III.1, III.2.1

[129] Stéphane Le Berre, Xavier Artusi, Roberto Miorelli, Ekaterina Iakovleva, and Pierre Calmon. Simulation and processing tools for the design and performance evaluation of FMC-TFM techniques. *AIP Conf. Proc.*, 2102(May), 2019. III.1

[130] Leonard Le Jeune, Sébastien Robert, P Dumas, A Membre, and Claire Prada. Adaptive ultrasonic imaging with the total focusing method for inspection of complex components immersed in water. In *AIP Conf. Proc.*, volume 1650, pages 1037–1046, 2015. III.1

[131] Corentin Mienard, Sébastien Robert, Roberto Miorelli, and Dominique Lesselier. Optimization algorithms for ultrasonic array imaging in homogeneous anisotropic steel components with unknown properties. *NDT & E Int.*, 116:102327, 2020. III.1

[132] Tuomas Koskinen, Iikka Virkkunen, Oskar Siljama, and Oskari Jessen-Juhler. The effect of different flaw data to machine learning powered ultrasonic inspection. *J. Nondestruct. Eval.*, 40(1):1–13, 2021. III.1

[133] Shamim Ahmed, Christophe Reboud, Pierre-Emile Lhuillier, Pierre Calmon, and Roberto Miorelli. An adaptive sampling strategy for quasi real time crack characterization on eddy current testing signals. *NDT & E Int.*, 103:154–165, 2019. III.1

[134] Peipei Zhu, Yuhua Cheng, Portia Banerjee, Antonello Tamburrino, and Yiming Deng. A novel machine learning model for eddy current testing with uncertainty. *NDT & E Int.*, 101:104–112, 2019. III.1

[135] Robert R Hughes and Bruce W Drinkwater. Exploring high-frequency eddy-current testing for sub-aperture defect characterisation using parametric-manifold mapping. *NDT & E Int.*, 124:102534, 2021. III.1

[136] Sergio Cantero-Chinchilla, Paul D Wilcox, and Anthony J Croxford. A deep learning based methodology for artefact identification and suppression with application to ultrasonic images. *NDT & E Int.*, 126:102575, 2022. III.1

[137] Iikka Virkkunen, Tuomas Koskinen, Oskari Jessen-Juhler, and Jari Rinta-Aho. Augmented ultrasonic data for machine learning. *J. Nondestruct. Eval.*, 40(1):1–11, 2021. III.1

[138] Gabriel Meseguer-Brocal and Geoffroy Peeters. Conditioned-U-Net: Introducing a control mechanism in the U-net for multiple source separations. *Proc. of the 20th ISMIR Conf., ISMIR 2019*, pages 159–165, 2019. III.2

[139] Liming Zhang, Wenbin Zhang, and Nathalie Japkowicz. Conditional-unet: A condition-aware deep model for coherent human activity recognition from wearables. In *2020 25th Int. Conf. on Pattern Pecognit. (ICPR)*, pages 5889–5896, 2021. III.2

[140] Lu Lu, Ming Dao, Punit Kumar, Upadrasta Ramamurty, George Em Karniadakis, and Subra Suresh. Extraction of mechanical properties of materials through deep learning from instrumented indentation. *Proc. Natl. Acad. Sci.*, 117:7052–7062, 2020. III.2

[141] Evgeny G. Bazulin. On the possibility of using the maximum entropy method in ultrasonic nondestructive testing for scatterer visualization from a set of echo signals. *Acoust. Phys.*, 59(2):210–227, March 2013. III.2.1

[142] Kombossé Sy, Philippe Brédif, Ekaterina Iakovleva, Olivier Roy, and Dominique Lesselier. Development of the specular echoes estimator to predict relevant modes for Total Focusing Method imaging. *NDT & E Int.*, 99:134–140, 09 2018. III.2.1, III.3

[143] Kombossé Sy, Philippe Brédif, Ekaterina Iakovleva, Olivier Roy, and Dominique Lesselier. Development of methods for the analysis of multi-mode tfm images. In *J. Phys.: Conf. Series*, volume 1017, page 012005. IOP Publishing, 2018. III.2.1

[144] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. III.2.2

[145] Jana Fragemann, Lynton Ardizzone, Jan Egger, and Jens Kleesiek. Review of disentanglement approaches for medical applications – towards solving the gordian knot of generative models in healthcare. *arXiv e-prints*, 2022. III.2.2

[146] Haris Iqbal. Harisiqbal88/plotneuralnet v1.0.0, December 2018. III.3, IV.3, IV.4

[147] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. IEEE Int. Conf. Comput. Vis.*, page 1510–1519, Venice, 09 2017. IEEE. III.2.2

[148] Liming Jiang, Bo Dai, Wayne Wu, and Chen Change Loy. Focal Frequency Loss for Image Reconstruction and Synthesis. *Proc. IEEE Int. Conf. Comput. Vis.*, pages 13899–13909, 2021. III.2.2, IV.3.3

[149] Benyamin Ghojogh, Mark Crowley, Fakhri Karray, and Ali Ghodsi. Adversarial Autoencoders. *e-J. Nondestr. Test.*, pages 577–596, 2023. IV.1

[150] Filippo Gatti and Didier Clouteau. Towards blending Physics-Based numerical simulations and seismic databases using Generative Adversarial Network. *Comput. Methods Appl. Mech. Eng.*, 372, dec 2020. IV.1, IV.3

[151] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Adv. Neural Inf. Process. Syst.*, volume 32, 2019. IV.1, IV.3, IV.3.1, IV.3.1, IV.3.2, IV.3.5

[152] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018. IV.1

[153] Jie Zhang, Bruce W. Drinkwater, Paul D. Wilcox, and Alan J. Hunter. Defect detection using ultrasonic arrays: The multi-mode total focusing method. *NDT & E Int.*, 43(2):123–133, 2010. IV.2.1

[154] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. IV.3

[155] Chunyuan Li, Hao Liu, Changyou Chen, Yunchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. ALICE: Towards understanding adversarial learning for joint distribution matching. In *Adv. Neural Inf. Process. Syst.*, volume 2017-Decem, pages 5496–5504. Neural information processing systems foundation, sep 2017. IV.3, IV.5

[156] Ilya Kavalerov, Wojciech Czaja, and Rama Chellappa. cGANs with Multi-Hinge Loss. *CoRR*, (2), 2019. IV.3

[157] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. IV.3

[158] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-Attention Generative Adversarial Networks. *arXiv e-prints*, may 2018. IV.3.1

[159] Alireza Makhzani. Implicit Autoencoders. *arXiv e-prints*, 2018. IV.3.1

[160] Siming Yan, Zhenpei Yang, Haoxiang Li, Li Guan, Hao Kang, Gang Hua, and Qixing Huang. IAE: Implicit Autoencoder for Point Cloud Self-supervised Representation Learning. *arXiv e-prints*, 2022. IV.3.1

[161] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015. IV.3.1.1

[162] Zhe Chen, Wenhai Wang, Enze Xie, Tong Lu, and Ping Luo. Towards Ultra-Resolution Neural Style Transfer via Thumbnail Instance Normalization. 2021. IV.3.1.1

[163] Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *Conf. Comput. Vis. Pattern Recognit.*, 2017-January:5967–5976, nov 2016. IV.3.2

[164] Takeru Miyato and Masanori Koyama. CGANs with projection discriminator. *6th Int. Conf. on Learn. Representations - Conf. Track Proc.*, 2018. IV.3.2

[165] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016. IV.3.3, IV.3.3, IV.5

[166] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. IV.3.3

[167] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. IV.3.5

[168] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017. IV.3.5

[169] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. IV.3.5

[170] Augustus Odena, Jacob Buckman, Catherine Olsson, Tom B. Brown, Christopher Olah, Colin Raffel, and Ian Goodfellow. Is generator conditioning causally related to gan performance?, 2018. IV.3.5

[171] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. IV.4.1

[172] Zhixin Pan and Prabhat Mishra. Fast approximate spectral normalization for robust deep neural networks. *CoRR*, abs/2103.13815, 2021. IV.6

[173] Qingyu Zhao, Zixuan Liu, Ehsan Adeli, and Kilian M Pohl. Longitudinal Self-Supervised Learning. Technical report, 2021. IV.6

[174] Qingyu Zhao, Zixuan Liu, Ehsan Adeli, and Kilian M Pohl. Longitudinal self-supervised learning. *Medical Image Analysis*, 71, 2021. IV.6, V.3.4

[175] Xiao Liu, Pedro Sanchez, Spyridon Thermos, Alison Q. O'Neil, and Sotirios A Tsaftaris. Learning Disentangled Representations in the Imaging Domain. *Medical Image Analysis*, 2021. IV.6, V.3.4

[176] Zinan Lin, Kiran K. Thekumparampi, Giulia Fanti, and Sewoong Oh. InfoGAN-CR and modelcentrality: Self-supervised model training and selection for disentangling gans. *37th International Conference on Machine Learning, ICML 2020*, PartF16814:6083–6095, 2020. IV.6

[177] Qingyu Zhao, Zixuan Liu, Ehsan Adeli, and Kilian M. Pohl. LSSL: longitudinal self-supervised learning. *CoRR*, abs/2006.06930, 2020. IV.6, V.2

[178] Vivek Nerlikar, Roberto Miorelli, Arnaud Recoquillay, and Oscar D'Almeida. A physics-embedded deep-learning framework for efficient multi-fidelity modeling applied to guided wave based structural health monitoring. *Ultrasonics [under review]*, 2023. IV.6

[179] Nir Shlezinger, Jay Whang, Yonina C. Eldar, and Alexandros G. Dimakis. Model-based deep learning, 2022. V.2.1

[180] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, Karen Willcox, and Steven Lee. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. 2 2019. V.2.7

[181] Yu Bai, Jiebo Luo, and Chang Wen Chen. Wavelet convolutional neural networks. *arXiv preprint arXiv:1805.08620*, 2018. V.3.2

[182] Jakub Zak, Anna Korzynska, Antonina Pater, and Lukasz Roszkowiak. Fourier transform layer: A proof of work in different training scenarios. *Applied Soft Computing*, 145:110607, 2023. V.3.2

[183] Alaa M. Alaa, Ahmed A. Mohamed, and Ehab El-Saadany. Generative time-series modeling with fourier flows. *arXiv preprint arXiv:2211.03470*, 2022. V.3.2

[184] Rui Guo, Tianyao Huang, Maokun Li, Haiyang Zhang, and Yonina C. Eldar. Physics embedded machine learning for electromagnetic data imaging, 2022. V.3.3, V.1

[185] Efron Bair and Trevor Hastie. Prediction by supervised principal components. *Journal of the American Statistical Association*, 98(462):651–661, 2003. V.3.4

[186] Robin Rombach, Andreas Blattmann, Dominik Lorenz, and Piotr Dosovitskiy. Stable diffusion models: A new approach to image generation. *arXiv preprint arXiv:2202.10258*, 2022. V.3.7, V.3

[187] OpenAI. Dall-e 2: An image generator that can create realistic images from text. *arXiv preprint arXiv:2201.07285*, 2022. V.3.7

[188] Dustin Boswell. Introduction to Support Vector Machines. 2002. VI.1

[189] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960. VI.3

[190] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012. VI.3.2

[191] Celina Jeffery. "To See the World in a Grain of Sand": Wolfgang Laib and the Aesthetics of Interpenetrability. *Religion and the Arts*, 17(1-2):57–73, 2013. VI.3.2

VI - Appendix - Complementary background for machine learning methods and techniques

VI.1 . Machine learning approaches based on kernel-based methods

Regression and classification problems can be affordable when a representative data set has enough instances (large N) for the class or label space $\mathcal{Y}$. Even when these two characteristics are fulfilled, sometimes the feature space $\mathcal{X} \subseteq \mathbb{R}^D$ can not be easily separated by a hyper-plane when using the linear methods enumerated in Fig. I.13.

A non-linear alternative is the kernel-based method. Kernel-based method relies on the 'kernel trick' when manifolds represent data[1]. Earth's surface is an example of a simple manifold: latitude and longitude points $p_i = (\text{latitude}, \text{longitude})$ can be positioned over its (approximately) spherical geometry. To better visualize the countries, the a priori $3D$ cloud of points may reach a flat representation in a sheet. The cloud on the sphere surface turns into a flat representation on a sheet of paper by a mapping function (e.g., the Mercator projection). p_i points are then mapped from a $3D$ to $2D$ representation.

Kernel techniques implement a mapping function for a data set that contains linear and nonlinear relations between its features. In contrast to the given example, kernel techniques are implemented for more complex point arrangements than just a regular grid in a sphere. For instance, a data set can be represented as complex geometry (or manifold). A sample $\mathbf{x}$ can be seen as a point in the manifold with non-linear relations between those features represented in the complex manifold. Instead of keeping a complex non-linear manifold in the rough original representation space $\mathcal{X}$, a kernel provides a mapping function represents data in a more suitable manifold. The resulting space may have a higher or lower dimension than the original space, the objective is to create an adequate representation so the points can be, for instance, separated easily by its classes, as in Fig. VII.1.

Considering two samples at $\mathcal{X}$, $\mathbf{a}$ and $\mathbf{b}$, the kernel $K(\mathbf{a}, \mathbf{b})$ is a function capable of performing dot product $\phi(\mathbf{a}) \cdot \phi(\mathbf{b})$ based on the original space vectors $\mathbf{a}$ and $\mathbf{b}$, being ϕ the mapping function from original manifold dimension to new representation of the data set. It is not necessary to fully define ϕ but only knows it exists. In this case, where K respect Mercer's theorem, we can define a kernel function in Eq.VI.1 where K is also a linear kernel implemented by Eq.VI.2.

$$K(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a}) \cdot \phi(\mathbf{b}) \tag{VI.1}$$

$$K(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \cdot \mathbf{b} \tag{VI.2}$$

This is an example of the Kernel Trick, where ϕ do not need to be defined and it can be expressed as a the inner product K in another space.

Other standard kernels with polynomial or Gaussian mapping functions apply the kernel trick. Regressors and classifiers like Supported Vector Machine (SVM are good candidates to implement a Kernel function [188]. The choice of the kernel depends on the original data manifold representation, sometime unknown. A kernel is commonly applied to data in the ML algorithm loss function to help the convergence and the learning of the task.

VI.2 . Machine learning based on DL

VI.2.1 . Perceptron Unit

[1] Manifolds describe a vast number of geometric surfaces in 3 or more dimensional spaces.

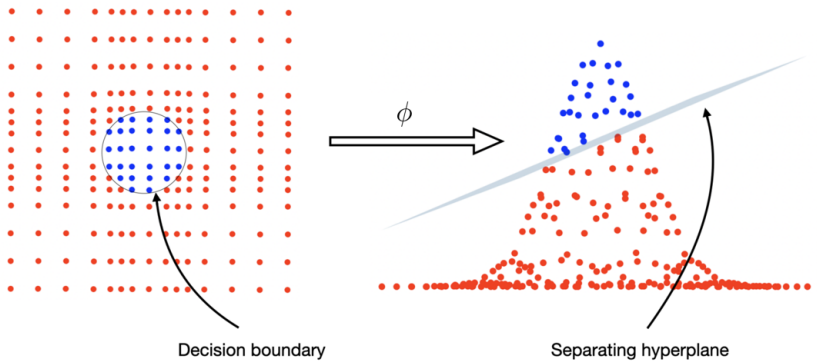


Figure VI.1. In the left, an example of a flat feature manifold for a $2D$ data set, features are represented by (x, y), and labeled by a class (colored in blue and red). In the right, the features are mapped by $\phi(x, y, z) \equiv (x, y, e^{-(x^2+y^2)})$ into a more suitable manifold, so a hyper-plane can separate the classes. In the original space of this example, a circular decision boundary shall be more complex to define the represented hyper-plane. Courtesy of hashpi.com [accessed September 05, 2023]

Consider a function f with i-dependent variables contained on a $\mathbf{x}_i$ array with $\mathbf{x}_i \in \mathbb{R}^D$ and an independent variable $\mathbf{y} = f(\mathbf{x})$, where f has parameters to find called weights ω_n and a bias b. f is represented in Fig.VI.2 as perceptron with its inputs and output.

A *perceptron* mimics the function of biological neurons in a simplified form. The weights represent the connection from the inputs to the cell core. An activation value is obtained by summing up the product of each input and its corresponding weight; this value is represented by the $\sum$ symbol. Note that the input also considers a bias value multiplied by 1. The bias helps to the perceptron to adjust its output by the approximation of the mean of the samples uses to find ω_n. The activation value feeds an activation function σ (e.g., a hyperbolic tangent as in Fig. VI.3). This function varies in each application, generating different outputs. For example, the use of identity in σ is a linear function with values from $-\infty$ to ∞.

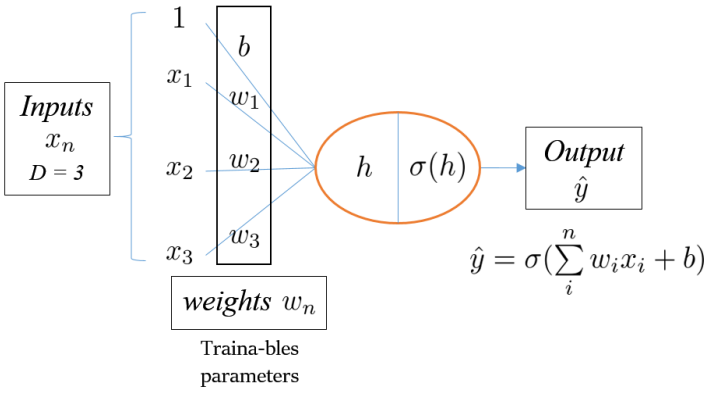


Figure VI.2. Schema presenting the perceptron unit. In this example, the number of features is 3, and the output is a single value $\hat{\mathbf{y}}$.

VI.2.2 . Layers and neural network architecture

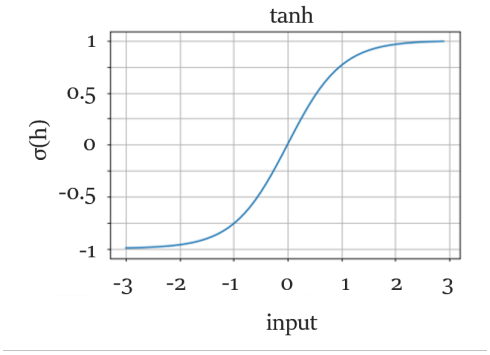


Figure VI.3. Example of an activation function: the hyperbolic tangent $\sigma(h^k)$. σ is placed at the output of a neuron activation, so $\hat{\mathbf{y}} = \sigma(\cdot)$.

Neurons are usually arranged in a set of layers, being $L \in (2, \infty)$ the number of layers. A *layer* is a group of neurons that shares the same input. A sequence of layers constitutes the neural network *architecture*. When the neural network is acyclic everywhere, they are often called feed-forward networks since they take an input to be propagated through the graph to generate an output. In the following definitions, we focus on ML methods that implement NNs since they are recurrently used later. However, ML methods do not only count in NN: polynomial regression, k-means clustering, among others, are examples of this statement.

VI.3 . Multi-layer perceptron or dense layer

The DNN extension of single-layer NN is done by groups of neurons that create a dense layer by connecting each input feature (x_i) to each neuron input that can be represented via the perceptron unit. Adding more layers and connecting each new layer input to each precedent layer output constitutes the basic architecture of a fully connected network (FC-NN) of L-layers, also referred as Multi-Layer Perceptron (MLP) in the bibliography. An example of this arrangement is shown in Fig. VI.4. Each connection in the FC-NN represents a weight (denoted as instance by ω_{ij} for the first layer) that is a learn-able parameter in the $\boldsymbol{\theta}$ space, where f_θ is represented by the FC-NN.

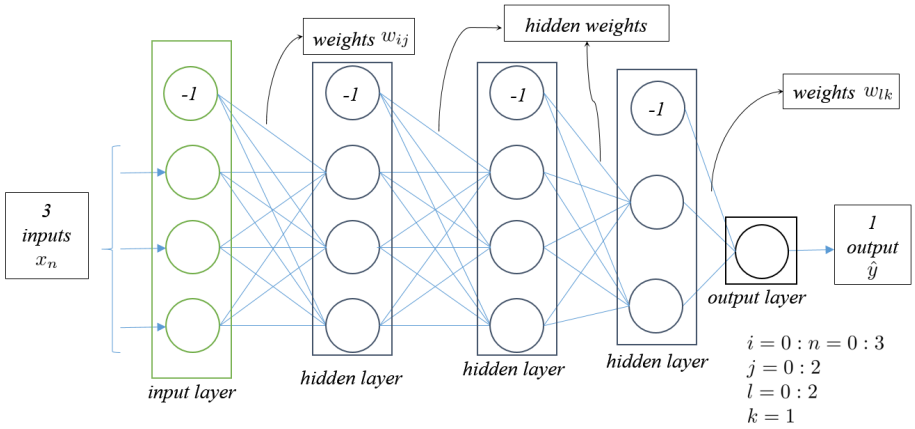


Figure VI.4. Fully connected network or multi-layer perceptron architecture with one hidden, three neurons in the input layer, and one neuron in the output layer.

The main constitutive layers of a FC-NN are the input layer with the same dimension of input samples vector $\mathbf{x}$. In the given sample (Fig. VI.4), the output has only one neuron, but in practical cases, this layer can have several output values.

FC-NNs are used for regression and classification tasks as data-driven models. The training stage commonly uses the *back-propagation* algorithm [189] explained in Annex VI.3.3. The objective of the optimization technique is to find $\boldsymbol{\theta}^*$. As initialization of the back-propagation, the weights (and bias) need to be set up for the first time in the network (*weight initialization*). There are many techniques to initialize $\boldsymbol{\theta}$, but the more common is to choose a distribution law and get the first values by sampling the distribution.

While performing the back-propagation, an output is estimated by the *forward propagation* on Eq.VI.3 and Eq.VI.4. In a feed-forward NN, the forward propagation happens when f_θ is evaluated for a given $\mathbf{x}$ at a fixed $\boldsymbol{\theta}$. A *loss function* $\mathcal{L}(\cdot)$ compares this output under a given criteria. $\mathcal{L}$ shall give a value to judge how near is f_θ to the optimal $\boldsymbol{\theta}^*$ for the defined task.

$$f_\theta(\mathbf{x}) = (\mathscr{L}_L \circ \sigma_L \circ \mathscr{L}_{l-1} \circ \sigma_{l-1} \circ ... \circ \mathscr{L}_1 \circ \sigma_1)(\mathbf{x}) \quad \text{with} \begin{cases} \theta : \text{set of weights and bias} \\ l = 1 : L \\ L : \text{number of layers on } f \end{cases}, \tag{VI.3}$$

where $\circ$ is the function composition operator. For instance, for two given functions h and g, the operator produced a composed function $(h \circ g)(x)$ where the function g is applied to the result of applying the function h to x. A single layer operation can be expressed as follows,

$$\mathscr{L}_l(\mathbf{x}^{l-1}) = \omega^l \mathbf{x}_f^{l-1} + \mathbf{b}^l \quad \text{with} \begin{cases} \omega : \text{weights vector for layer } \ell_k \\ b : \text{bias vector for } \ell_k \\ \mathbf{x}_f^{l-1} : \text{output from presedent layer} \\ (\cdot)_f : \text{inner feature (layer output) for } \mathbf{x} \end{cases}. \tag{VI.4}$$

VI.3.1 . Loss functions

Before introducing the back-propagation algorithm, the *loss function* $\mathcal{L}_{\boldsymbol{\theta}}(\cdot)$ needs to be defined. In order to train a FC-NN (or any NN), a metric is defined to calculate the model error based on the computed output. The loss function may be defined by one metric or by adding several metrics to be optimized. This function is minimized for each epoch.

Different metrics exist to quantify the error of the NN output. Regression tasks used metrics like Mean Square Error (MSE) or Mean Absolute Error (MAE) (Eq. VI.5), among others. Suppose the estimator has suitable complexity regarding the number of layers and trainable parameters. In that case, the task of the NN shall improve when the loss function value decreases (Fig. VI.5) when training on $\mathcal{D}$.

Similarly, classification problems using functions such as logistic regression (Eq. VI.6), cross-entropy (Eq. VI.7) or hinge function. This kind of loss function measures if the estimator can infer the actual class of the input $\mathbf{x}$. As instance, a logistic regression loss is a suitable metric for two classes problem ($C = 2$); while the cross-entropy function is a general function for multi-classes tasks.

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{y}, \hat{\mathbf{y}}) = E[(\hat{\mathbf{y}} - \mathbf{y})^2] \quad \text{with} \left\{ \hat{\mathbf{y}} : \text{estimation vector of the batch.} \right. \tag{VI.5}$$

$$\mathcal{L}_\theta(\mathbf{y}, \hat{\mathbf{y}}) = max(0, \hat{\mathbf{y}} \cdot \mathbf{y}) \tag{VI.6}$$

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i^C \hat{\mathbf{y}}_i log(\mathbf{y}_i) \quad \text{with} \left\{ \hat{\mathbf{y}} : \text{binary class indicators for the example.} \right. \tag{VI.7}$$

Several metrics for different tasks are present in the bibliography. They will be defined in future sections when implemented for particular applications.

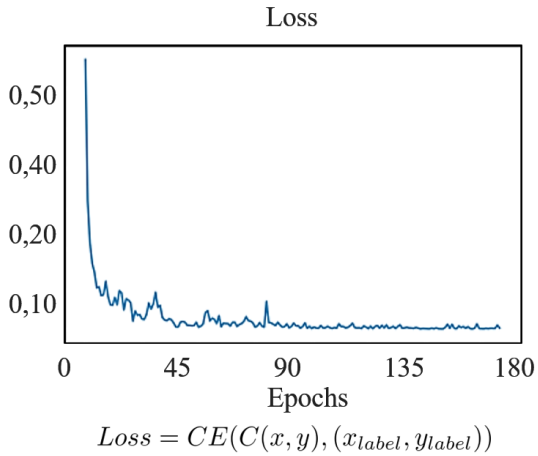


Figure VI.5. Example of cross-entropy loss function decreasing on 180 epochs for a classification problem of two classes.

VI.3.2 . Hyper-parameters in NNs

Besides the trainable parameters in a FC-NN (weights and bias), it is necessary to define an architecture to fix the number of layers (L), number of neurons per layer, or activation functions at each layer. Extensible to any NN, the listed elements are called *hyper-parameters* of a NN. They must be fixed before training, and take no part in the back-propagation optimization.

How to choose the best set of hyper-parameters in a given NN (e.g., FC-NN) is an active research field in the ML community. A vast ensemble of techniques are proposed in the literature to try to give a solution to this problem ranging from the trial-and-error to Bayesian optimization, among others [190, 191].

VI.3.3 . Back-propagation algorithm

Now that the loss function was introduced, the back-propagation as the optimization algorithm for a NN can be described. Back-propagation is an iterative optimization algorithm that employs gradient descent to adjust neural network weights in the direction opposite to the gradient of the performance function. The aim is to find a set of weights that minimizes the loss function, thus solving the learning problem.

The NN $f_{\boldsymbol{\theta}}$ is expected to improve on each epoch at the task. $\mathcal{L}$ is a suitable metric that tends to 0 when $f_{\boldsymbol{\theta}}$ is better for the given task. For this, the trainable parameters in the algorithm are fitted (or optimized) with the back-propagation of the output error from $\mathcal{L}$.

From here, we specify $\mathcal{L}$ and f to simplify the equations for a concise example. f is considered an acyclic graph composed of dense layers. Defined MSE as a metric and an arbitrary derivable activation function σ, the parameters $\boldsymbol{\theta}_o = \{\boldsymbol{\theta}_o, \boldsymbol{\theta}_h\}$ for a FC-NN with one hidden layer and an unique output are $\boldsymbol{\theta}_h = w_{ij} | i = 0, 1, ..., m_h - 1; j = 0, 1, ..., n_h - 1$ and $\boldsymbol{\theta}_o = w_{jk} | j = 0, 1, ..., n_o - 1; k = 0, 1, ..., m_o - 1$. m_h is the number of inputs for the first layer, $n_h - 1 = n_o - 1$ is the number of hidden neurons, and $m_o = 1$ is the number of outputs. A simplified equation used to update $\boldsymbol{\theta}$ is:

$$\delta w_{ij}^h = \epsilon_{lr} \frac{\delta \mathcal{L}}{\delta w_{ij}^h} \quad \text{where: } h \text{ : hidden layer weights,}$$

$$\delta w_{jk}^o = \epsilon_{lr} \frac{\delta \mathcal{L}}{\delta w_{jk}^o} \quad \text{where: } o \text{ : output layer weights,}$$

$$w(t+1) = w(t) + \delta w, \tag{VI.8}$$

where $\epsilon_{lr} < 1$ is the *learning rate*, w_{ij} is the parameter to update the connection of the i-neuron to the j-neuron between the input and hidden layer, and w_{jk} is the connection of the j-neuron to the k-neuron between the hidden and output layer. $w(t+1)$ is the new value of any weight after the back-propagation at the end of the epoch $t \subset \mathbb{Z}^+$, when the correspondent δw is added to the initial value $w(t)$.

The value of δw in the Eq.VI.8 is needed to derive the loss function and the activation function. The resulting update equation takes ϵ_{lr}, the scalar input from the previous layer x_i or x_j, and a Δ gradient that represents the portion of the error due to the concerning weight, as follows:

$$w_{ij}(t+1) = w_{ij} + \epsilon_{lr}\Delta_j^h x_i,$$

$$w_{jk}(t+1) = w_{jk} + \epsilon_{lr}\Delta_k^o x_j. \tag{VI.9}$$

The algorithm starts from the output error to calculate Δ_k and Δ_j in Eq. VI.9, the algorithm starts from the output error. In this case, with one output, only a Δ_k is computed with the following equation:

$$\Delta_k^o = [y - \sigma^o(h^o)]\frac{\partial \sigma^o(h^o)}{\partial h^o} \tag{VI.10}$$

Note that the derivative is defined by h^o, which is the activation value of the function σ previously calculated as the sum of the weight and inputs for the concerning neuron (see Fig. VI.2). It is worth noting that $\sigma^o(h^o)$ is the predicted output before denoted as $\hat{y}$.

Now, with Δ_k calculated with the derivative of σ and the output error, the algorithm is capable of computing Δ_j by:

$$\Delta_j^h = \sum_k (\Delta_k^o w_{jk}^o)\frac{\partial \sigma^h(h_j^h)}{\partial h_j^h} \tag{VI.11}$$

This last equation reveals the reason for the name given to the algorithm. The error from the output is propagated backward to the previous layers by a Δ_k^o weighted by the w_{jk}^o connections. The derivative of the activation function in VI.10 and VI.11 gives the algorithm the direction (positive or negative) and the proportion of change for each weight. The learning rate regulates the general speed of change through all updates. When implementing more hidden layers and outputs, the general case can be inferred similarly.

VI.4 . Principal component analysis for dimensionality reduction

Principal component analysis [47] uses the concept of explained variance to measure the loss of information during the dimensional reduction. The technique performs an eigenvalue extraction of a correlation matrix $\boldsymbol{V}$ built for each feature and through all samples in the data set. Diagonal matrix $\boldsymbol{D}$ is expressed in terms of the correlation matrix as:

$$\boldsymbol{V}\boldsymbol{v}_i = \sigma_i^2 \boldsymbol{D}\boldsymbol{v}_i \tag{VI.12}$$

where σ_i^2 is the eigenvalues of the correlation matrix and $\boldsymbol{v}_i$ the eigenvectors associated. The principal direction $\boldsymbol{c}_i$ of data are obtained from the eigenvectors. Relation between the vector and its principal direction is expressed in Eq. VI.13.

$$\boldsymbol{c}_i = \boldsymbol{X}\boldsymbol{v}_i \tag{VI.13}$$

$\boldsymbol{X}_{n\times m}$ is a matrix containing the m original representation vectors (number of samples) with n features for each vector; $\boldsymbol{c_i}$ is the data set's principal direction.

The vector $\boldsymbol{c}_i$ associated with the larger eigenvalue represents a principal direction of the set in its original representation (before reducing its dimension), where the maximum possible data variance is included. The second principal direction is orthogonal to the first and next vectors, which have the same characteristic. Posterior projection of the data set in those directions creates a reduced and finite representation space. An instance's principal component (PC) is the value of the vector x_i projected in the principal direction.

This technique allows selecting only the first n_z-eigenvectors to represent an instance in the associated directions. The more variance required to be included in the new representation, the more principal directions must be included in the reduced space.

VII - Appendix - Parametric spatial transformer and feature-wise linear modulation layers on conditional AutoEncoder architecture

This section aims to illustrate the functioning of the parametric Spatial Transformer (pST) and Feature-wise Linear Modulation (FiLM) layers in the context of a classic AutoEncoder (AE) architecture. The objective is to comprehend how the AE utilizes input for generation and why it can be arbitrarily altered during the prediction (test) phase.

Fig. VII.1 illustrates the generation of Eddy Current Testing (ECT) signals for different inputs. It is crucial to emphasize that in this appendix, the AE was trained with identical input and output, allowing the generation of diverse images by simply changing the input image while maintaining the same parameter vector for all presented generations. This follows conventional conditional AE training.

This initial demonstration indicates that a bias is introduced by the input during prediction. Moreover, the most suitable input is the image corresponding to the intended output. This is useful if we intend to use the AE as a meta model, as we are not supposed to know the output before generating it on a meta model.

Chapter II showcases an AE-like architecture trained differently: the input is fixed at an arbitrary value, and the neural network is then trained. The choice of fixing the input is based on the observation in Fig. VII.1, complemented by Fig. VII.2. The latter demonstrates that pST and FiLM act to change the extracted features based on the input vector. However, especially for the pST affine transformation, it is evident that the actions of the conditional layer in the encoder do not retain useful image features to generate the outputs. One can imagine that the primary features are used in encoding and decoding to create a new ECT image; however, the experiments conducted here show that the encoder does not require input features but only the parameter vector.

In conclusion, coding is primarily influenced by the parameter vector rather than the input image. Additionally, a bias effect is introduced by changing the input. Subsequent training with fixed noise, as shown in Chapter II, confirms this. Improvement was observed when the input is fixed from the training. For instance, R-squared reported an improvement from 0.887 to 0.920, resulting in better image generation.

These AE-like architectures can be viewed as key components of recently developed diffusion models (refer to Chapter V, section V.3.7 for more details). In other words, the architecture can generate new images from noise guided by the simulation parameter. Additionally, the architecture provides a reduced number of training parameters that competes with other deeper architecture (e.g., an dense layer based architecture).

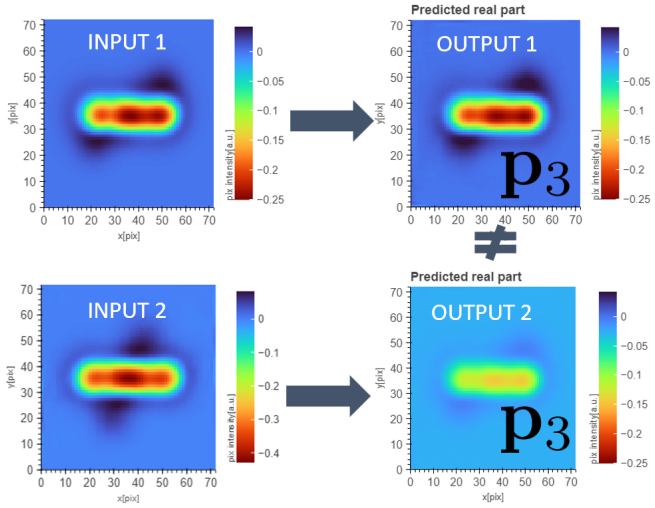


Figure VII.1. Each row depicts the same generation conditioned by $\mathbf{p}_3$ after training the architecture as a conventional AE. The first row takes the Ground Truth (GT) for $\mathbf{p}_3$as input, showcasing a straightforward reconstruction with high quality. In the second row, the input images are not labeled by $\mathbf{p}_3$ and differ slightly in terms of features from the $\mathbf{p}_3$ image. In this scenario, we observe the bias introduced by the input.

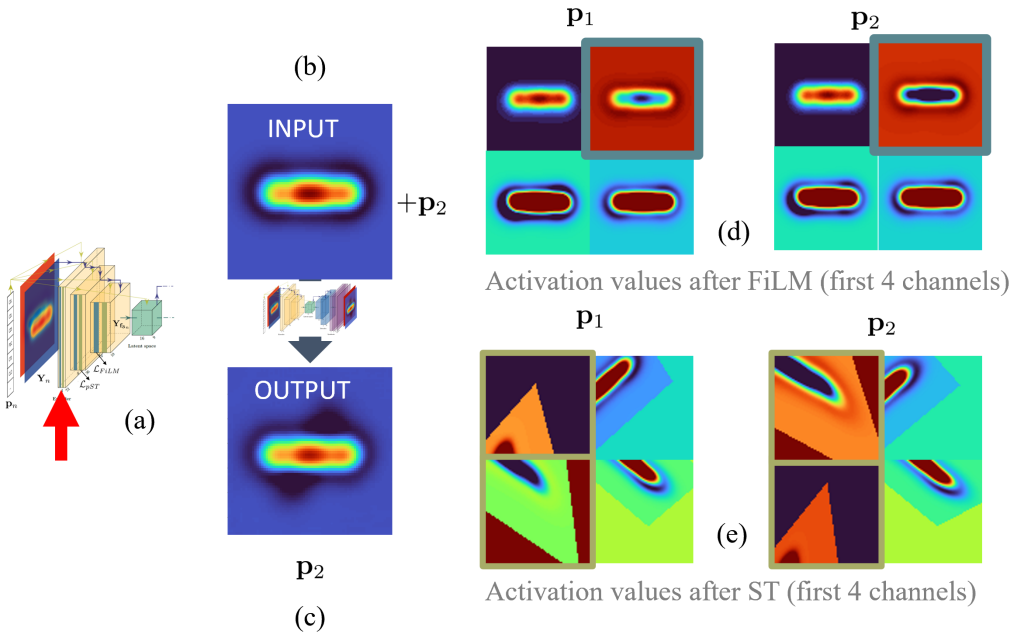
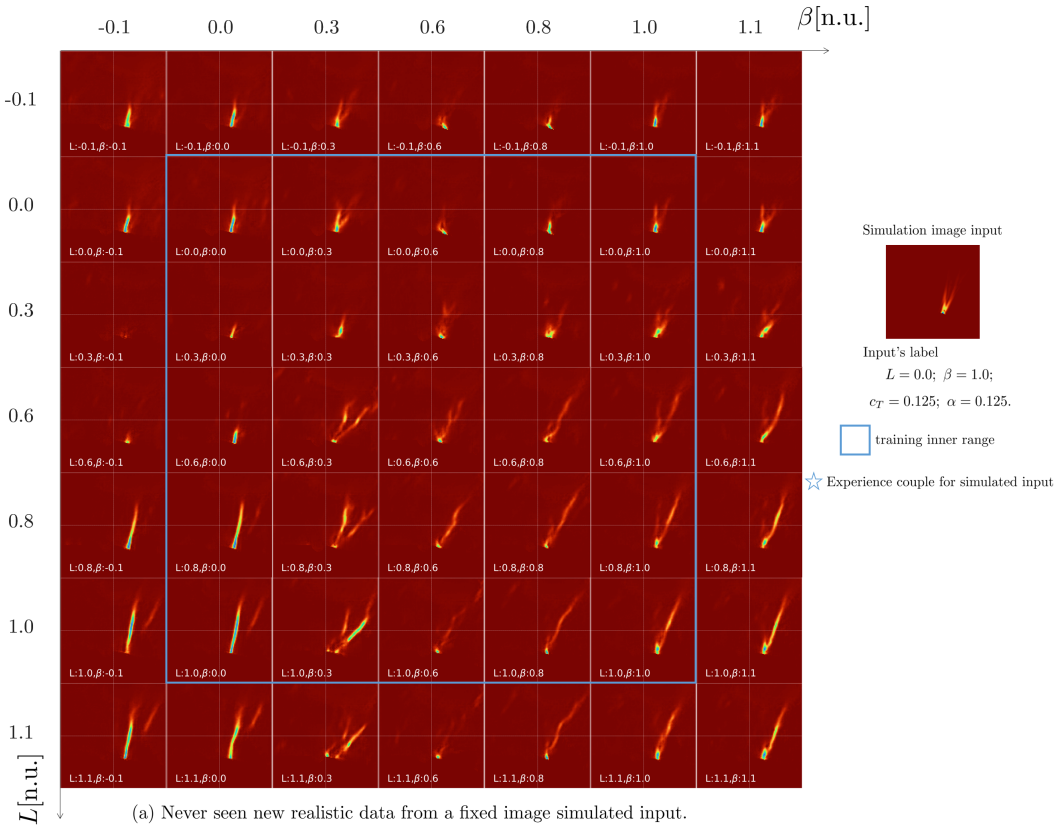


Figure VII.2. We present the conditioned features extracted for ECT image generation using an architecture trained as a conventional AE. In (a), the red arrow indicates the considered layer for extraction: the first pST and FiLM layer. The generation is computed by using an image labeled by $\mathbf{p}_1$ (b) as input and a conditioned parameter vector $\mathbf{p}_2$. The expected generation is showed in (c), representing the Ground Truth (GT) for $\mathbf{p}_2$. In (d), we illustrate how FiLM extracts features during generation for two cases. The four images labeled $\mathbf{p}_1$ depict the resulting activation when a simple reconstruction is intended. The four images labeled $\mathbf{p}_2$ represent the resulting activation when the input is still the image labeled by $\mathbf{p}_1$, but the input vector is $\mathbf{p}_2$. It can be observed that the flaw signature changes in amplitude driven by $\beta(\mathbf{p}_2)$ and $\gamma(\mathbf{p}_2)$ from equation II.6. Similarity, (e) demonstrates the effect of the pST layer for both reconstruction and generation. Both (d) and (e) show that the encoder does not retain useful features for either the reconstruction of $\mathbf{p}_1$nor the generation of $\mathbf{p}_2$. Instead, the encoder utilizes the input image to map it to coding, considering only $\mathbf{p}_2$. This experiment, along with the AE-like training in Chapter II, indicates that the input features are not genuinely useful for generating a new ECT image guided by $\mathbf{p}$. The images from the layer activation are normalized per instance, allowing the observation of differences in pixel intensities. Each of the four activation maps is transformed into an RGB format in the range of 0 to 255 (Min-Max).

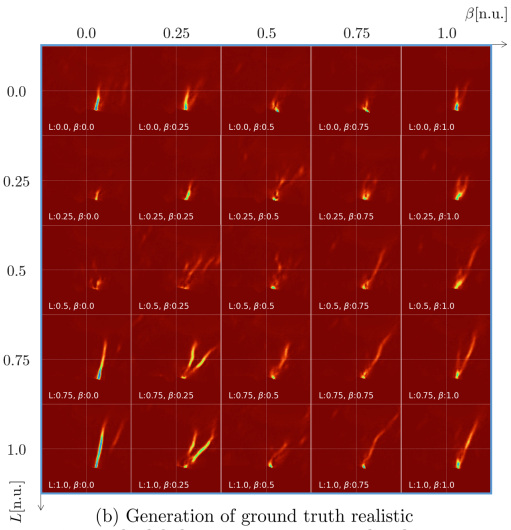
VIII - Appendix Realistic data generation by trained tailored cU-Net

Figures above demonstrate the Deep Neural Network (DNN) meta-model potential on generating realistic Multi modal Total Focus Method (M-TFM) images. Comparison to ground truth is made where possible. Parameter variations $\mathbf{p}$ are employed for predictions. Consistency with physics is observed in terms of wave celerity and back-wall angle. However, accuracy decreases when altering flaw height and angle due to the limited diversity of the four-flaw experimental data set. Consequently, extrapolation performance cannot be deemed sufficiently accurate. Two reconstruction modes highlight distinct impacts of flaw geometry and reconstruction parameters.

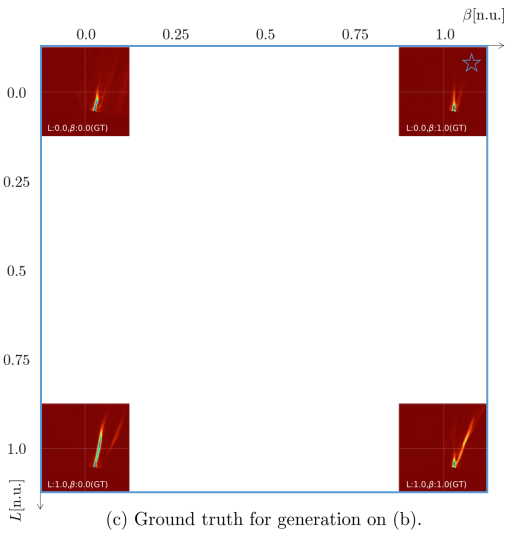
DNN realistic generation by regression over flaw geomety parameters $\boldsymbol{L}$ and $\boldsymbol{\beta}$. Example of **TT-T** M-TFM mode.
c_T and α are fixed during regression



(a) Never seen new realistic data from a fixed image simulated input.

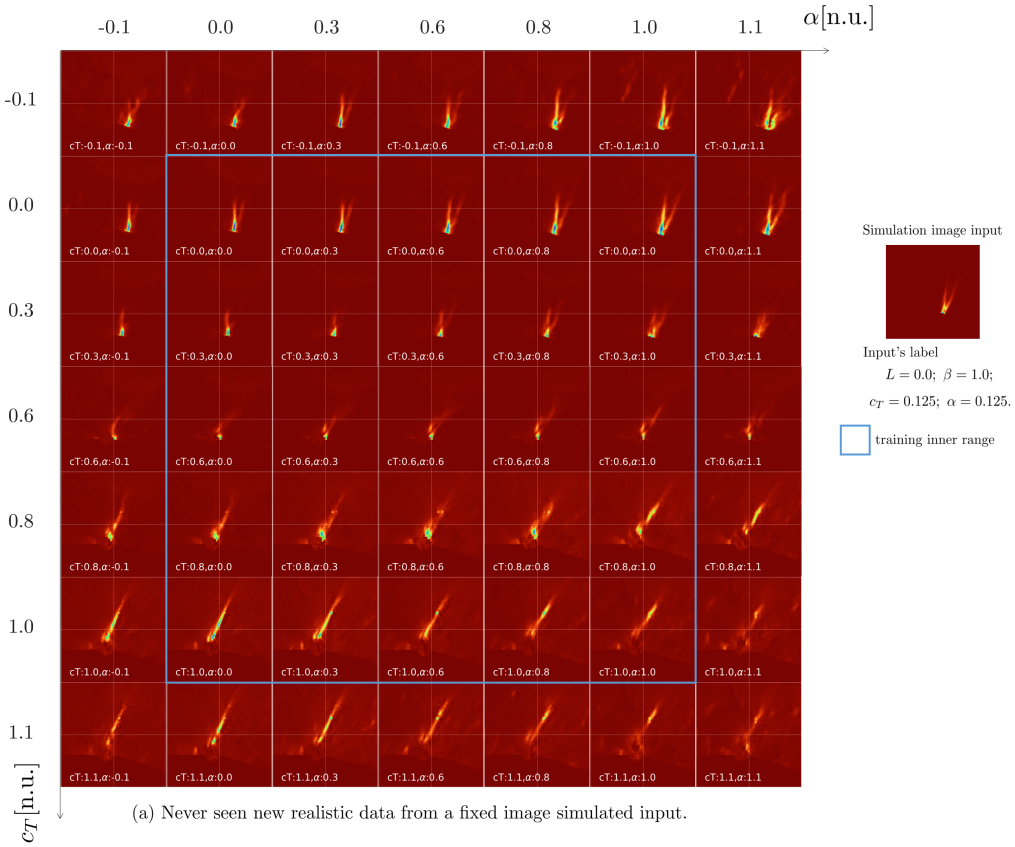


(b) Generation of ground truth realistic images by label regression over a simulated input.

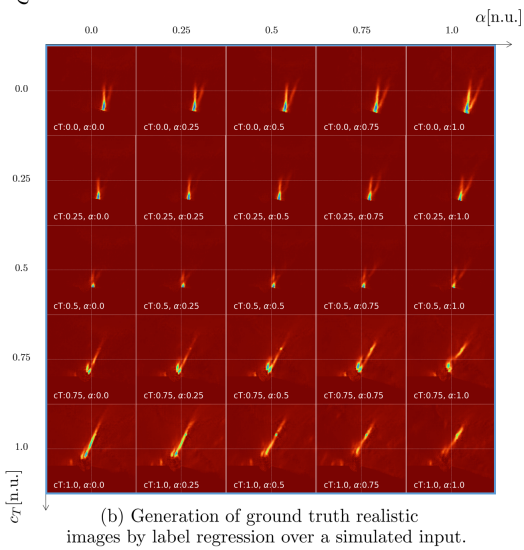


(c) Ground truth for generation on (b).

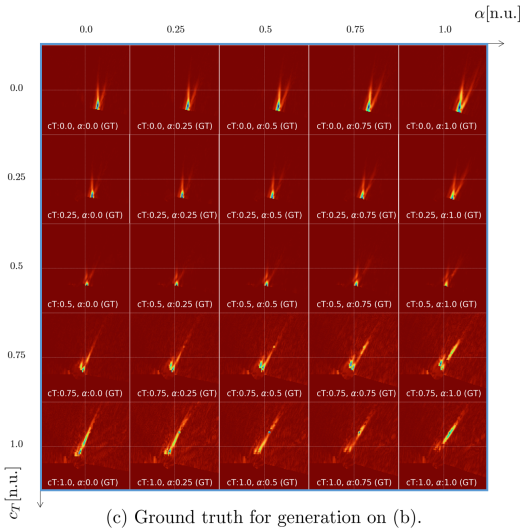
DNN realistic generation by regression over M-TFM reconstruction parameters $\boldsymbol{c_T}$ and $\boldsymbol{\alpha}$. Example of **TT-T** M-TFM mode.
L and β are fixed during regression



(a) Never seen new realistic data from a fixed image simulated input.

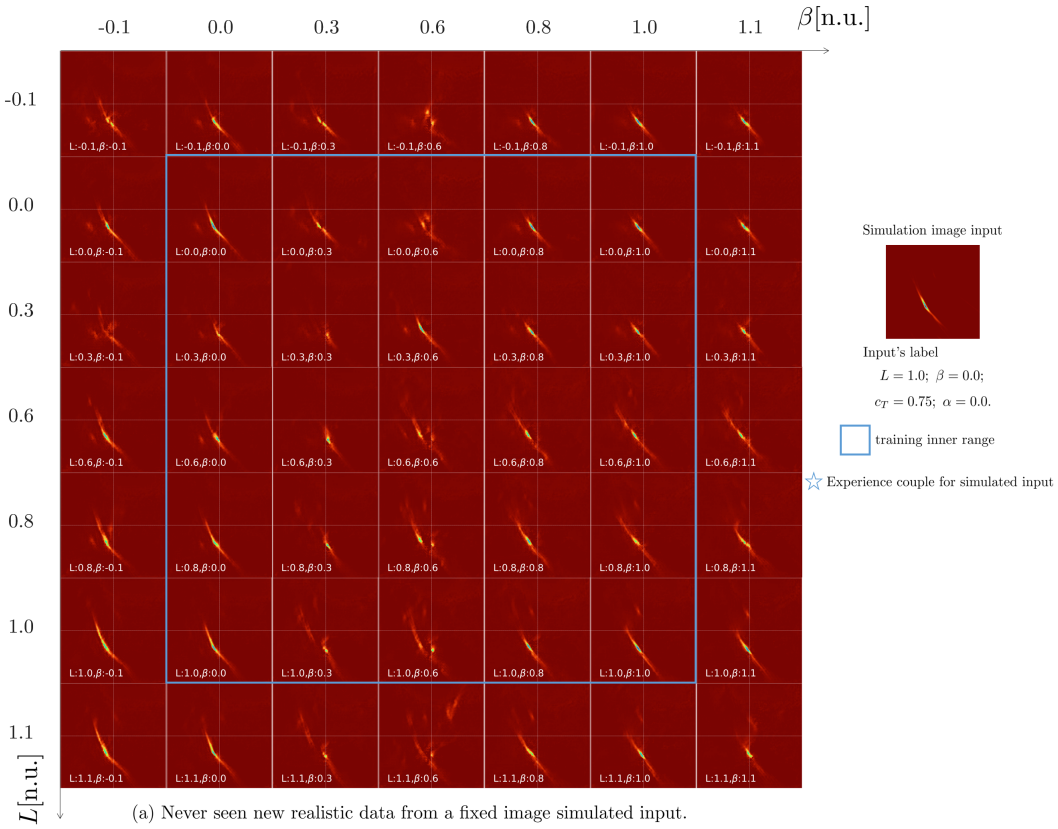


(b) Generation of ground truth realistic images by label regression over a simulated input.

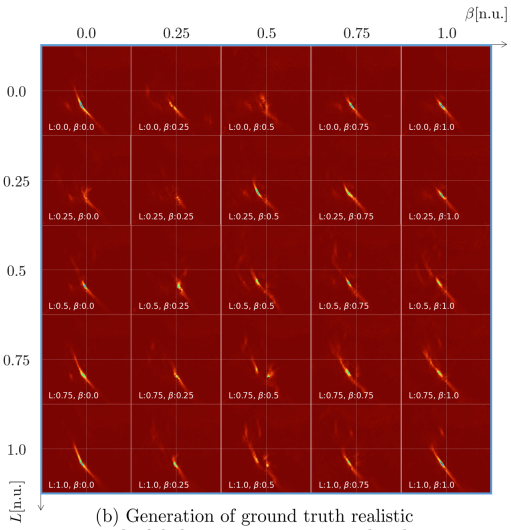


(c) Ground truth for generation on (b).

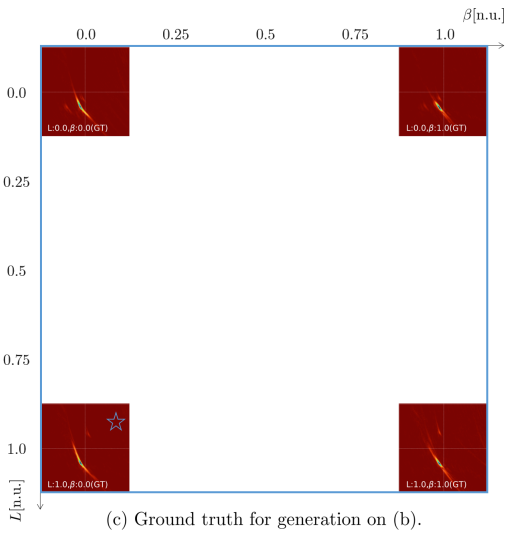
DNN realistic generation by regression over flaw geomety parameters $\boldsymbol{L}$ and $\boldsymbol{\beta}$. Example of **TT** M-TFM mode.
c_T and α are fixed during regression



(a) Never seen new realistic data from a fixed image simulated input.

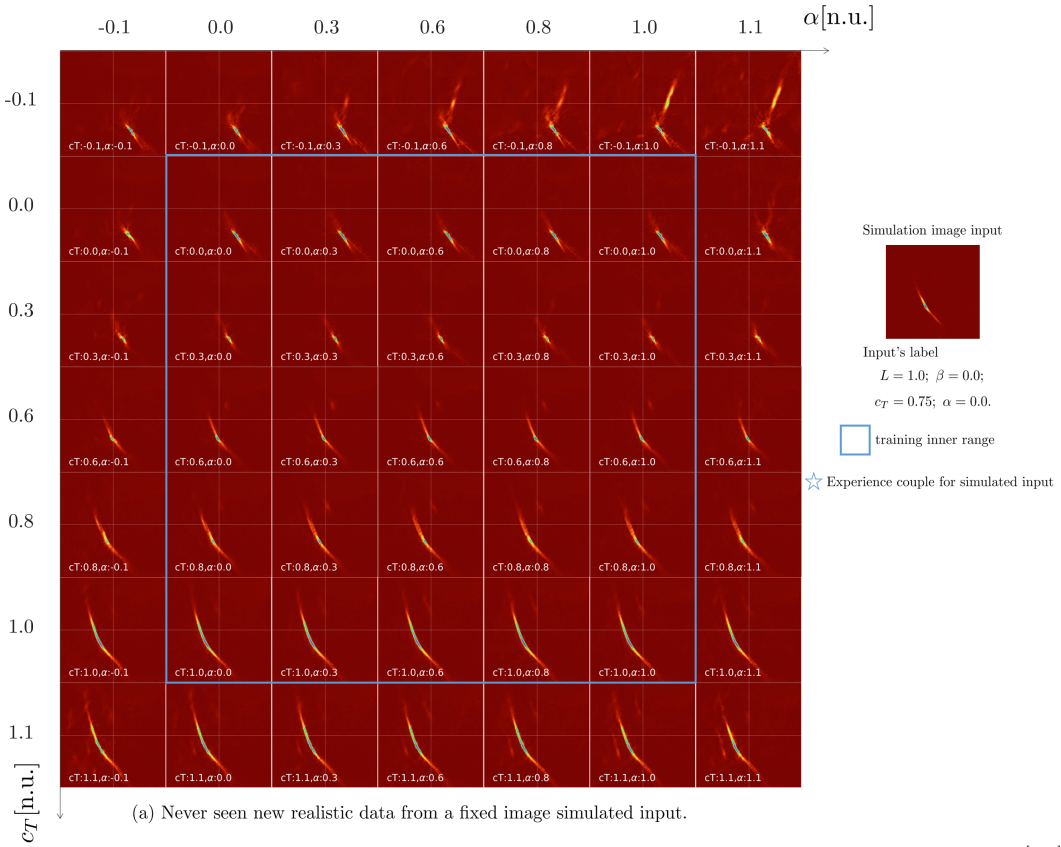


(b) Generation of ground truth realistic images by label regression over a simulated input.

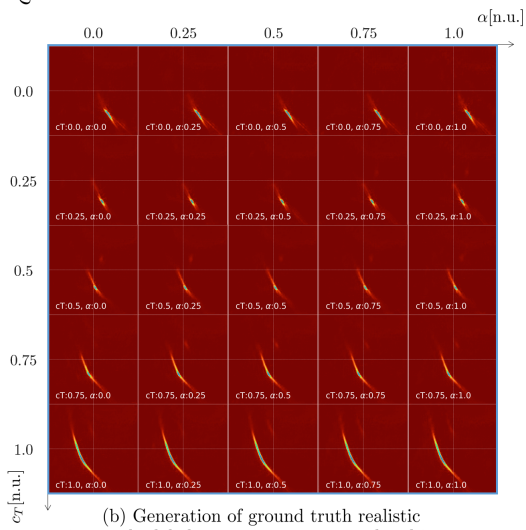


(c) Ground truth for generation on (b).

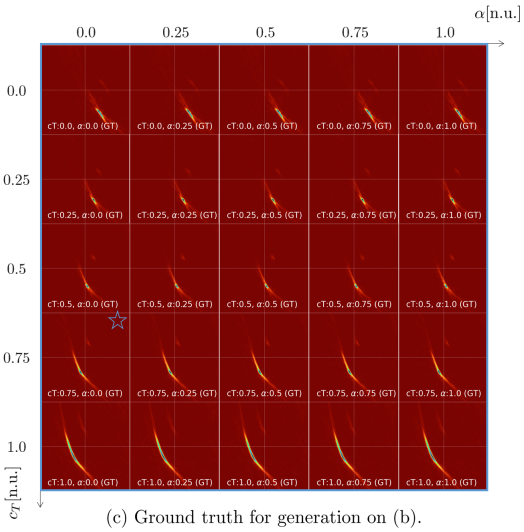
DNN realistic generation by regression over M-TFM reconstruction parameters $\boldsymbol{c_T}$ and $\boldsymbol{\alpha}$. Example of **TT** M-TFM mode.
L and β are fixed during regression



(a) Never seen new realistic data from a fixed image simulated input.



(b) Generation of ground truth realistic images by label regression over a simulated input.



(c) Ground truth for generation on (b).