StyleGAN-based machining digital twin for smart manufacturing

A thesis submitted to the University of Sheffield for the degree of Doctor of Philosophy

Evgeny Zotov

Department of Automatic Control and Systems Engineering

April 2022

Dedicated to the unknown
for not all being unknowable

Acknowledgements

I would like to thank my supervisor Professor Visakan Kadirkamanathan for continuously putting his trust in my work and for offering invaluable guidance without limiting the freedom of thought that produced this work.

I would also like to express my gratitude to Professors Geoffrey E. Hinton and Kenneth O. Stanley for producing the works that initially ignited my interest in the neural network domain.

Special thanks go to my friends and family for their high tolerance of my stubborn desire to introduce them to the alien concepts that are mostly not present in their world. And an additional very special thank you is reserved for my daughter for keeping asking the question "Why?".

Abstract

Manufacturing enterprises are challenged to remain competitive due to the increasing demand for greater product variability and quality, intensifying complexity of the production processes, as well as a drive for sustainable manufacturing and the increasing regulatory impact resulting in high labour and energy costs. Consolidated around the discussion of Industry 4.0, the efficient and effective solutions to these challenges lie outside the mainstream production methods. One of the drivers of transition towards the novel manufacturing paradigm is the technological modernisation of the production processes motivated by the increasing availability of computational capacities.

Manufacturing digitalisation is a critical part of the transition towards Industry 4.0. Digital twin plays a significant role as the instrument that enables digital access to precise real-time information about physical objects and supports the optimisation of the related processes through conversion of the big data associated with them into actionable information. A number of frameworks and conceptual models has been proposed in the research literature that addresses the requirements and benefits of digital twins, yet their applications are explored to a lesser extent.

The work presented in this thesis aims to make a proposition that considers the novel challenges introduced for data analysis in the presence of heterogeneous and dynamic cyber-physical systems in Industry 4.0. In this thesis a time-domain machining vibration model based on a generative adversarial network (GAN) is proposed as a digital twin component. The developed conditional StyleGAN architecture enables (1) the extraction of knowledge from existing models and (2) a data-driven simulation applicable for production process optimisation. A novel solution to the challenges in GAN analysis is then developed, where the comparison of maps of generative accuracy and sensitivity reveals patterns of similarity between these metrics.

The proposed simulation model is further extended to reuse the knowledge extracted from a source model and adapt it to a given target environment, enabling the elicitation of information from both physics-based and data-driven solutions. This approach is implemented as a novel domain adaptation algorithm based on the GAN model: CycleStyleGAN. The architecture is validated in an experimental scenario that aims to replicate a real-world manufacturing knowledge transfer problem. The experiment shows that the transferred information enables the reduction of the required target domain data by one order of magnitude.

The thesis thus builds a strong case for a StyleGAN-based digital twin to be developed to support practical implementation of technologies paving the road towards the target state of Industry 4.0.

Contents

List of Figures

Chapter 1

Introduction

1.1 Research motivation and context

1.1.1 Industry 4.0 and the digital twin

The 4th industrial revolution, i.e. the strategic vision of transition to Industry 4.0, draws a path to a totally customisable production with viable single-item customisable batch production, just-in-time execution and high resource-efficiency including zero waste and right first time fabrication. Advances along this path are believed to be feasible as a result of pervasive digitalisation throughout the industry, spanning from the shop-floor to the whole supply chain and to the users of the end-products [45]. The steps to achieve this state include development of advanced sensing systems capable of in-process and embedded metrology that enable high-accuracy representation of the manufacturing systems' state. The effectiveness of processing of the prospective metrological data relies on novel manufacturing analytics methodologies. The efficiency of such analysis depends on the manufacturing informatics technologies that should support the communication latency and throughput requirements implied by the functional requirements of the underlying analytics systems. Therefore, several IT challenges introduced by Industry 4.0 have to be addressed to make the implementation of smart factory technologies more accessible and to thus accelerate the digitalisation rate throughout industry, as summarised in fig. 1.1 [88].

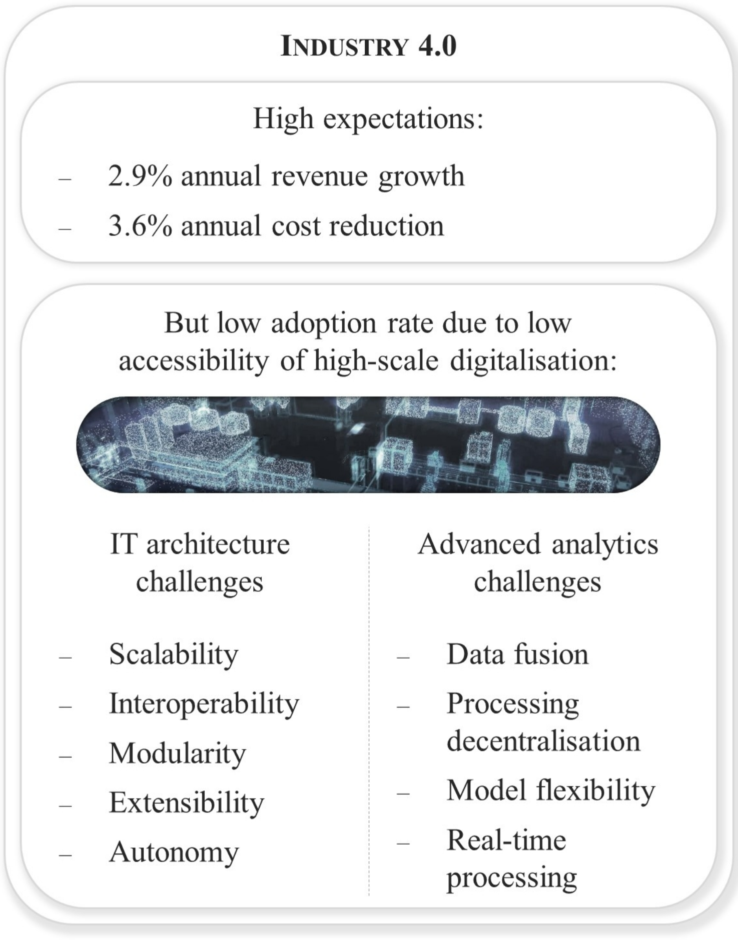

INDUSTRY 4.0
High expectations:
– 2.9% annual revenue growth
– 3.6% annual cost reduction
But low adoption rate due to low accessibility of high-scale digitalisation:
IT architecture challenges
– Scalability
– Interoperability
– Modularity
– Extensibility
– Autonomy
Advanced analytics challenges
– Data fusion
– Processing decentralisation
– Model flexibility
– Real-time processing

Figure 1.1: **Industry 4.0 analytics and informatics challenges summary.**

Total factory digitalisation is being made possible by the technologies emerging from the research fields of big data, cyber-physical systems (CPS) and industrial internet of things (IIoT) [64]. The implementation of these technologies is becoming increasingly accessible as the big data technology stack has matured over the last couple of decades, offering many open-source and proprietary software tools supporting its deployment on commodity hardware [2]. As a result, IIoT implementation became feasible [77] and one anticipates seeing the industry shift rapidly towards widespread adoption of IIoT technologies as companies compete to disrupt the market or to maintain their position on it by reaping the performance and efficiency benefits. This would be a significant step in the evolution towards complete CPS integrated end-to-end through the value chain. Despite notable progress in development of the strategic vision of Industry 4.0 and increasing affordability of hardware and software solutions, actual implementation of smart factory technologies on manufacturing shop-floors remains relatively low, especially among small and medium enterprises [112]. Researchers and industry experts attribute this fact to the high complexity of execution, long return on investment periods and high investment costs [50, 112].

Precise representation of physical objects or processes in the digital realm is researched within the digital twin domain. Definition of the digital twin has varied throughout its existence: from the aircraft-oriented definition originally introduced in the aerospace domain research [96] to the modern concept of digital twin as a complete digital recreation of whole ecosystems [8]. The main aspect common to all the proposed scales is the absolute information equivalence between the perfect digital and the physical twins, implying that any interaction with one is mirrored in another [40]. Development of digital twins is an important step of the digitalisation process, as the unification of digital and physical data within a single virtual object enables significant efficiency improvements across multiple stages of the object's life cycle [111].

A holistic digital twin requires integration of multiple interconnected models and metrology tools that capture the different aspects of the complete system. The progress towards this goal would thus be followed in stages with gradual development of digital twin components, including infrastructure, monitoring and predictive systems. Only after significant research advancement in these directions would the digital twin become usable as a decision-making tool capable of delivering the robust efficiency improvements promised by technology visionaries [40, 78].

1.1.2 Simulation and data-driven modelling

Simulation modelling is a widely used technique employed in the verification of engineering designs and evaluation of their functionality and performance. By utilising the known information about the product geometry, its material characteristics and operating conditions, the virtual recreation of its operation environment enables analysis of the product's behaviour. A product can be similarly studied during a manufacturing process, and its feature characteristics both after and during processing can be predicted [84].

The enhanced efficiency is going to be increasingly attractive to the manufacturers of high-value products, as simulations enable a shifting of the physical experimentation costs to virtual analysis. And the increased flexibility shall enable agile product design, which is a prerequisite for agile manufacturing. In both cases a significant reduction of waste, cost and lead times is to be expected [29].

The computational constraints limiting the usefulness of simulation modelling in previous decades is significantly compensated for by the modern hardware and software advances. Thus, simulations have become a critical tool utilised in the analysis of the various aspects related to the machining processes over the last several decades [104]. For example, experimentally validated time-domain simulation models are used in the dynamic modelling of machining process stability, enabling real-time chatter suppression for increased machining stability [4], as well as in the generation of 3D milled surface profiles [15]. The finite element method (FEM) is a widely-used approach for simulation of physical processes, under which a complex problem (or geometry) is subdivided into its smaller parts for which the approximations of interactions can be computed via relatively simple equations. Examples of FEM applications in machining include the simulation of the chip formation and the cutting forces in precision machining [1] and the simulation of the tool stress and temperature in high-speed milling [82]. Over the years the physics-based FEM models have been specialised to niche applications, such as for 3D simulation of titanium alloy micro-end milling chip flow and tool wear [113] or for simulation of the 3D interactions of various composite material drilling techniques and their effect on composite delamination [98]. By enabling the off-production virtual experimentation, the simulations thus aid the practitioners during the configuration of the cutting conditions and the tool path planning prior to the execution of the machining process [5]. Further refinement of the established simulation modelling techniques with hybrid data-driven and knowledge-based approaches is an ongoing research topic [39].

The growing complexity of manufacturing processes and products has been identified as a stable trend by the end of the previous century [125]. The variety of error sources and their dynamic nature significantly distort the model predictions and present a considerable challenge that has been and still is thoroughly studied by manufacturing researchers [26]. These sources include the material uncertainties (such as its workability, shear stress or deviations from material specifications) and machining uncertainties (for example, tool wear and runout, machine geometry and thermal errors) [68, 126]. These production process-related uncertainties are propagated and magnified throughout the potential measurement errors arising from the measurement strategy choice, fixturing and environment variability, as well as from measurement tool and software errors [31, 84]. The result is the chaotic variability of the behaviour of the modelled processes under different conditions and in heterogeneous environments.

Physics-based analytics models tend to achieve high accuracy rates, but have several drawbacks that can become blocking factors for implementation of a digital twin. On one hand, in an interconnected CPS environment interactions between the components introduce very high complexity of the modelled phenomena. On the other hand, the incremental character of modular development and the fluid module composition cause an almost constant stream of changes in the system [80]. This and the utilisation of empirically estimated parameters that approximate some of the unobserved factors in physics-based models imply that a given model has to be manually adapted to every new environment or scenario it is deployed in. Data-driven modelling addresses this issue by making use of the big data produced by the various manufacturer's CPSs and automating the modelling process, thus aligning the digital twin state with the evolutionary changes in the modelled systems [75]. Therefore, despite the wide success of physics-based simulation models for prediction of abnormal conditions during milling and turning processes, data-driven simulation methods offer greater flexibility at adapting to a broader range of conditions, including dynamically changing ones [32]. Moreover, increasing amount of effort is directed towards fusing the physics-based knowledge acquired by the previous generations with the information extractable from the new digital data sources [114].

1.1.3 Artificial neural networks in manufacturing

Development of efficient and flexible data-driven simulation models of physical manufacturing processes is an important step towards CPS digitalisation in general, and particularly to wide adoption of digital twins throughout the industry. Artificial neural network (ANN) is a machine learning model inspired by the information flow structure of a brain's neurons. Scaled versions of these networks labelled as deep learning models [63] have shown increasingly impressive state-of-art results on many data-driven problems [52]. Among the many works that have resonated throughout the community are the first AI to beat the grand champion in the game of Go [102, 103] and the protein folding model capable of atomic accuracy predictions of protein structures, even ones that have never been seen before [53]. Deep learning models have attracted a lot of attention from the manufacturing research community for their wide applicability to problems from different domains and of different scopes [86]. ANNs have been utilised in manufacturing applications ranging from conventional machine health monitoring for fault detection and classification [124, 129] to the more advanced predictive analytics for machine health prognosis [51] to the real-time product quality monitoring based on acoustic emission signals in additive manufacturing [99].

The proposed applications of ANNs to machining domain problems include chatter prediction for turning processes via the classification of the points on a stability lobe diagram [18]. For milling chatter analysis a use case is available concerning the prediction of chatter frequency and of the critical depth of cut at which the regenerative chatter occurs [81]. ANNs have also been used for machined surface roughness prediction for various machining process configurations [87, 118]. And a review paper on smart machining highlights the research works that have applied ANNs to the predictive modelling of surface quality and machining parameter optimisation in laser, abrasive water jet and electric discharge machining [57].

Generative adversarial network (GAN) is a type of ANN architecture based on a competitive minimax game between two ANNs: the generator that learns to produce artificial data samples and the discriminator that learns to identify fake data samples [38]. The data distribution produced by the generator network thus approaches a representation of the true data which can be directly utilised for simulation of the process underlying the data. In the few years since its inception the approach has been extended in multiple directions. The research on various neural network architectures for the generator and the discriminator introduced the deep convolutional model DCGAN for unsupervised image feature extraction and picture generation [90], as well as the SeqGAN model for reinforcement learning-based sequence generation validated on text and music synthesis tasks [127]. The drive for the generation of higher fidelity imagery meant that the output resolution of the computationally expensive GANs had to be increased while maintaining the tractability of the calculations. This has been addressed by the authors of StackGAN via an ensemble of low-resolution GAN paired with a super-resolution GAN that showed reasonable performance in 256x256 image generation [128]. The authors of BigGAN further scale up the GAN model to accommodate 512x512 image synthesis capability [14]. Reviews of the GAN training approaches have produced notable solutions, such as the progressive growing of the GAN topologies [55], where the authors propose to incrementally increase the model output from 4x4 to 1024x1024 during training, thus pushing the synthetic image resolution even further. The lack of control over the original GAN's outputs led the research efforts to the development of methods for the conditioning of the GAN models by additional inputs or outputs. Initially proposed in Conditional GAN [73], where the image generation was successfully tied to input labels via supervised learning, successive work showed the unsupervised conditional training approach based on information theory with InfoGAN [16]. Furthermore, the ss-InfoGAN extension of this approach has showed how the human understanding of the relationship between the labels and the synthetic output conditioned on these labels can be enforced within the model with minimal supervision [105].

Recently GANs have advanced the state of the art in various domains. For example, in neural audio synthesis WaveGAN adapts the DCGAN model mentioned above for audio sequence generation [24], and GANSynth is developed for the generation of high-quality audio spectrograms [28]. The authors of [54] include a GAN into an ensemble model for speech reconstruction from a compact representation, and the authors of VoiceGAN adopt style transfer techniques for voice impersonation, i.e. the translation of speech style onto content [35]. RelGAN has advanced the state of the art in text generation, while also introducing a model configuration parameter for controlling the ratio of the synthesised samples' diversity to their quality [79]. Finally, a most notable advance in simulation of realistic high-resolution (1024x1024) images of human faces has been achieved via the Style-

GAN architecture that has adapted the style transfer methods for the GAN's generator network [56]. With most of GAN studies focussed on image-generation, the generation of time-domain signals with GANs remains a very narrow field of research. Some examples can be found within the publications on healthcare, such as the radiation dose optimisation for lung cancer patients' radiotherapy [115] and the generation of time-domain medical data [30]. Wind and solar energy output prediction models based on GANs have been proposed for the renewable energy scenario generation [17]. An attempt at the synthesis of classical music samples is also available with the C-RNN-GAN [74].

Data augmentation aimed at the support of a primary classification model currently dominates the research agenda on GANs in manufacturing, with the very similar use of the generator network applied for generation of rare fault samples for planetary gearbox fault diagnosis [42, 122] or for wafer fabrication defect recognition [120], for generation of missing data for energy consumption monitoring [121] and in a digital twin-based prognostics and health monitoring system [11]. This is also evidenced by multiple review papers touching on ANN and GAN applications in manufacturing, where the only identified use case for GAN models is data augmentation [51, 59, 89, 119]. An increasing plethora of works is being published in the recent years that propose GAN solutions for enhancing the classification accuracy of machinery fault detection through data augmentation, as evidenced in the review papers covering just the deep learning research on the machinery fault diagnosis topic [66]. Adoption of a GAN's generator as a primary instrument for manufacturing problems are discussed within image-generation GAN research considering generative material design [110] and sample super-resolution [3]. Additionally, a very recent anomaly detection approach was proposed in [22] via inversion of the generator, i.e. the authors considered the optimisation of the generator input with the rest of the network fixed for a given data sample. The generative accuracy of the sample produced by the GAN this way is suggested as a measure of the abnormality of the given data sample. Thus, the research on the generative function of GANs for manufacturing process simulation remains largely unexplored.

1.1.4 Knowledge transfer in manufacturing

The main limitation of the data-driven modelling is its requirement for data. At the current stage of industrial digitalisation it is likely that a data-driven method would lack the range of empirical data variability necessary for capturing the underlying process behaviour. Moreover, the data-driven model would not be able to reason from first principles unless merged into a hybrid system with the incorporation of a physics-based model. It is thus likely that a training regime based on hybrid dataset comprised of the experimental data and the data obtained from an existing model would in practice be employed at first. Such hybrid approaches are actively investigated by the research community [39], but their implementation is limited by the access to the underlying models of the physics-based tools used in industry. The proprietary modelling software rarely provides flexible integration access to its outputs, even less so to its internal procedures. Nevertheless, the information contained within such models is of great value, usually reflected in the price tag of the proprietary modelling software. The extraction of this information in a reusable form is one of the applications of the methods found in the knowledge transfer research domain [83] and is the main topic of this work.

Knowledge transfer for data analytics deals with the problem of heterogeneous or non-stationary environments, where a model effective in some environment requires adaptation to new or changed conditions to remain accurate [9]. In the context of manufacturing this usually implies that the target domain data is very limited, thus the need for efficient knowledge transfer from data-abundant domains. Knowledge transfer is usually regarded as an approach for information transfer between machine learning methods [83]. But for simulation modelling the current work shows that the source model does not strictly necessary have to be a learned model, as knowledge is extractable from physics-based simulation models using a GAN.

A recent review by [9] groups the knowledge transfer methods into two groups: incremental learning and domain adaptation. The main difference between the approaches in these groups is the discarding of the source domain data during incremental learning, as only the source domain knowledge encoded via a trained model is carried over to the target domain training phase [36]. Domain adaptation, on the contrary, implies that the source domain data is at least partially available and used to learn a mapping between the source and the target domains [34, 123].

Several recent domain adaptation methods are inspired by the adversarial interactions within GANs, as reflected in the terminology used to describe these techniques: adversarial domain adaptation. They are usually categorised as either feature-level [69, 108] or pixel-level adversarial domain adaptation approaches [12, 101] with some of the recent works merging the two in hybrid adversarial domain adaptation models, e.g. [13].

GAN-based approaches are widely popular in image processing domain adaptation, with examples available in unsupervised image transformation via the CoGAN, which is trained to add or remove certain attributes to the images of faces or to transform colour images into depth images [69]; CycleGAN [131], an image-to-image translation network that utilises mirrored duplex-GAN architecture for image style transfer between two domains; and the CyCADA model that has introduced cycle-consistency loss for image segmentation and image-to-image translation [47]. A recent work proposed a technique for knowledge propagation and knowledge sharing between the classes in conditional GANs via the sharing of batch normalisation parameters [97]. Another recently published paper adopted the attention mechanism that has been gaining popularity recently for improving the accuracy of text-to-image translation task [109].

Current research on domain adaptation GAN approaches in manufacturing follows the same pattern, as the manufacturing GAN research in general: while popular in the image translation domain, the use of GANs for time-series domain adaptation problems is relatively limited. The application of digital twin simulations in the context of knowledge extraction for Industry 4.0 is thus an even less studied area than the application of GANs for manufacturing process simulation [130].

1.2 Research aims and contributions

1.2.1 Thesis aims and objectives

The research presented in this thesis aims to address the implementation challenges that the manufacturing industry is facing in its transition to a smart manufacturing paradigm throughout the 4th industrial revolution, also known as Industry 4.0. This work is focused on information technology innovation that enables manufacturing companies to transform their facilities into "smart factories". Thus, this dissertation pursues the following objectives:

1. Extraction of the manufacturing process knowledge contained within the existing industrial modelling tools and its reuse in a digital twin.

2. Data-driven adaptation of the information, either extracted from or embedded in the existing models, to new environments with limited available data.

Chapter 2 of this thesis introduces the reader to the basic philosophical and technical background relevant for understanding the research methods proposed in the subsequent chapters.

Chapter 3 introduces the novel conditional StyleGAN model as a digital twin component. This module is aimed at capturing the conditional distribution of a machining vibration signal, thus enabling the configurable simulation of the underlying process. The performance and reliability of this model are evaluated using a novel sensitivity analysis approach. And the applications of the proposed method for manufacturing process optimisation and for knowledge extraction are explored.

The subsequent chapter 4 narrows the inspected machining simulation modelling problem towards the issue of the practical applicability of the StyleGAN model. The adaptability of the data-driven simulation model to specific environment conditions under data limitation constraint is investigated, and a novel GAN architecture is proposed to address the identified issues. The novel CycleStyleGAN model is implemented as a domain adaptation method aimed at the facilitation of knowledge transfer from a data-rich domain. This approach is validated in an experiment designed to approximate a real-world manufacturing deployment that shows the almost tenfold data volume optimisation capability.

1.2.2 Research contributions

The work presented henceforth has partly been previously made public within the following research papers:

- The StyleGAN model presented in chapter 3 has initially been brought forward within the conference paper:

  E. Zotov, A. Tiwari, and V. Kadirkamanathan. Towards a digital twin with generative adversarial network modelling of machining vibration. In L. Iliadis, P.P. Angelov, C. Jayne, and E. Pimenidis, editors, Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference, pages 190–201, Cham, 2020.

- The subsequent journal publication has discussed the StyleGAN modelling analysis:

E. Zotov, A. Tiwari, and V. Kadirkamanathan. Conditional StyleGAN modelling and analysis for a machining digital twin. Integrated Computer-Aided Engineering, pages 1–17, jul 2021.

- And chapter 4 is based on the following journal publication:

  E. Zotov and V. Kadirkamanathan. CycleStyleGAN-based knowledge transfer for a machining digital twin. Frontiers in Artificial Intelligence, 4, 2021.

The data used in the experiments presented in this thesis is freely available at GitLab:

- Chapter 3: `https://gitlab.com/EZotoff/conditional-stylegan-digital-twin`
- Chapter 4: `https://gitlab.com/EZotoff/cyclestylegan-based-knowledge-transfer-for-a-machining-digital-twin`

Chapter 2

Artificial neural network models

2.1 Artificial intelligence and deep learning

The philosophical thought of humanity for seemingly the entirety of its existence has considered the problem of consciousness. Yet, the millennia-long debate yields poor explanation not only of the origin of consciousness, but even of its existence. The acquisition of consciousness or intelligence by inanimate entities is a popular narrative represented in the mythos of different peoples: from the early stories about gods bringing matter to life or craftsmen producing living automata, through the medieval concept of a golem and the Enlightenment period's Frankenstein, to the modern computer-based artificial intelligence (AI). The common-sense contemporary understanding of an AI is highly influenced by the seminal works of Alan Turing that first proposed the concept that we now call software [19], followed by the widely popular Turing Test introduced for the purpose of identification of a human-like intelligence in a machine [116]. While AI as a field of research was formulated based on the information theory and cybernetics research areas at the Dartmouth College Artificial Intelligence Conference [71] in 1956, the application of this idea found more success with the fiction authors, rather than research scientists. The development of a general-purpose AI, like the ones portrayed in the western culture's cinematography, for example, in "2001: A Space Odyssey" (1968) or "Alien" (1979), is to date a work in progress surrounded by debate on its feasibility, ethics and existential consequences. While the philosophical arguments about human-level AI are now mostly treated as a prerogative of public speakers, the science and industry communities have narrowed the AI development effort and found ways to train the machines to effectively perform specific tasks towards the end of the 20th century. Machine learning methods that mostly existed since the inception of AI have suddenly resurged, driven by the increasing availability of computational resources. Within the first two decades of the 21st century, the breakthrough research significantly pushed the performance of specialised AI in different areas. To name a few, human-level accuracy in handwriting recognition [20] (based on the MNIST dataset) and super-human performance in image classification [43] or playing the game of Go [102] has been achieved. The vast majority of these technological marvels utilise an Artificial Neural Network (ANN) as the substrate that captures the knowledge necessary to perform the specialised task that the network is trained in. The study of large-scale ANNs has since been regarded as a separate subfield of machine learning, named deep learning owing to the concept of depth used in the formulation of ANN architectures. The following sections of this chapter provide an overview of the basic concepts that constitute the elements of ANNs. This overview is intentionally kept simple in order to accommodate both non-technical readers and those unfamiliar with the presented topics. Works that provide the complete detail of the relevant concepts are referenced alongside their condensed introductions.

2.2 Artificial neural networks

2.2.1 Artificial neurons

Perceptron, introduced in [92] and later expanded upon in [93], is a simple processing unit that converts some input vector $x = \{x_1, .., x_n\}$ of length n into a binary output y,

$$y = \begin{cases} 1 \text{ if } \sum w_i x_i + b > 0 \\ 0 \text{ otherwise} \end{cases}, \tag{2.1}$$

by taking the weighted and biased sum of its inputs ($\sum w_i x_i + b$), where w_i are the scaling factors (weights) applied to these inputs and b, called bias, is a constant shift added to the resulting weighted sum, and evaluating whether the result is positive or not. In the former case the perceptron is excited, outputting 1, or it outputs 0 in the latter case. Itself being inspired by the mechanism of a biological brain's neuron excitation, this module influenced the development of the basic unit of the ANNs — the artificial neuron. Formulated generally, the artificial neuron outputs an excitation value y,

$$y = f(\sum w_i x_i + b), \tag{2.2}$$

by taking the weighted and biased sum of the neuron's inputs, $\sum w_i x_i + b$, and applying some activation function f to the result.

A thorough review of the advantages and disadvantages of the different activation functions depending on their role in a ANN's architecture is a research topic of its own and is explored in the published research, e.g. [23]. Among the simplest functions is the step function that can be used to specify the neuron's behaviour to that of the perceptron:

$$f(z) = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ if } z < 0 \end{cases}. \tag{2.3}$$

While usable for manually constructed boolean logic functions [72], the main problem with the application of this activation function in the training artificial neural networks is its non-differentiability, because most gradient-based training methods require the functions within the neural network to be differentiable. This issue and the available solutions are addressed below in section 2.2.3.

Initial solution that aimed to replicate the step function, but in a smooth differentiable form, is the sigmoid activation:

$$f(z) = \frac{1}{1+e^{-z}}, \tag{2.4}$$

which is likewise bound by the range $(0, 1)$. Due to this property, it is often used when an output represents a ratio or a percentage. A modification of the sigmoid activation function that attempts to improve the ANN trainability and also adds a potentially desirable property of approximating the identity function near $z = 0$ is the tangent hyperbolic function:

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{2.5}$$

Another activation function, arguably the most widely used at the moment of writing, is the rectified linear unit (ReLU) [37]:

$$f(z) = max(0, z). \tag{2.6}$$

The idea behind its formulation again draws inspiration from the hypothesised functioning of the human brain, particularly the fact that its activity is mostly sparse, which means that most, 95% – 99%, of the brain's neurons remain inactive during regular activities [65]. The ReLU thus tries to emulate this sparsity by nullifying all the non-positive inputs it receives. The authors argue that the neural network is thus lent the opportunity of learning separable, disentangled representations of the data it processes, due to the direct ability of silencing or "turning off" individual neurons. This ability can make the network output insensitive to small changes in some of its inputs, which is harder to achieve with, for example, sigmoid neurons, where each input node is connected to all the downstream neurons.

Despite the empirical effectiveness of the sparsity introduced by ReLU [23], its use entails a drawback originating from the training of the ANNs based on back-propagation of the network's errors. As will be evident from the discussion in section 2.2.3, the constant output of a ReLU for negative inputs provides the network with no information about the direction towards potentially optimal network weights. This means that some data samples, for which a neurons inputs turn out to be negative, stop advancing the ANN training towards optimum. And therefore, if for some ReLU neuron all the data samples result in zero activation, this neuron is effectively "dead", as its output shall never be different from zero. To prevent the potential non-trainability of neurons while preserving the capability for developing a sparse representation of the data, an extension of the ReLU is proposed - the leaky ReLU [70]:

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{otherwise} \end{cases}, \tag{2.7}$$

where $0 < \alpha < 1$ is the leakiness coefficient that determines the slope of the activation function for negative inputs. By using a small value for α, the negative values that the neuron receives are no longer translated into a constant output, but are still closely approximate to the sparsity-inducing zero. The non-constant output implies that the gradient is still computable for the negative input range, which is expected to improve the network trainability.

The selection of optimal activation functions seems to depend on the specific task or data structures that an ANN tries to learn. To the best of authors knowledge, the choice of suitable activation functions for a specific implementation is currently based on an empirical evaluation of the AI engineer building the said network, as no robust and complete comparison of the activation functions' performance has been published to date. Some data suggests that the ReLU produces the best classification accuracy [23], but this fact requires robust verification across a wider range of ANN architectures and network sizes, as well as for other modelling tasks. An alternative approach is the inclusion of the activation functions types as a hyperparameter in the neural network architecture optimisation process. It must be noted though, that for a deep ANN this means that each of the many network layers could be parametrised with a different activation function. Thus, despite its potential for improving the effectiveness and efficiency of the trained model, the multiplicative increase of the possible network architecture variations implies that an expert-based approach for activation function selection would significantly reduce the requirements for computational resources' availability. Therefore, in this thesis the choice of activation functions is guided by the previous works that introduced the used ANN modules.

2.2.2 Network layers

A single neuron on its own does not possess the computational capacity to perform anything extraordinary, unlike an ensemble of numerous neurons forming the layers that constitute a ANN. For example, one of its simplest forms is a three-layer neural network:

$$y = f_y(z) = f_y(f_z(x)), \tag{2.8}$$

where x is the networks' inputs, z the result produced by the so-called hidden layers that process the network's inputs to be passed to the last, output layer that transforms them into y. And f_z and f_y are activation functions used in the hidden and the output layers respectively. A graphic representation found on fig. 2.1 helps the comprehension of this concept by presenting a basic example of an ANN with two inputs, one hidden layer with three neurons and a single output. Such a network architecture is referred to as the multi-layer perceptron. Each

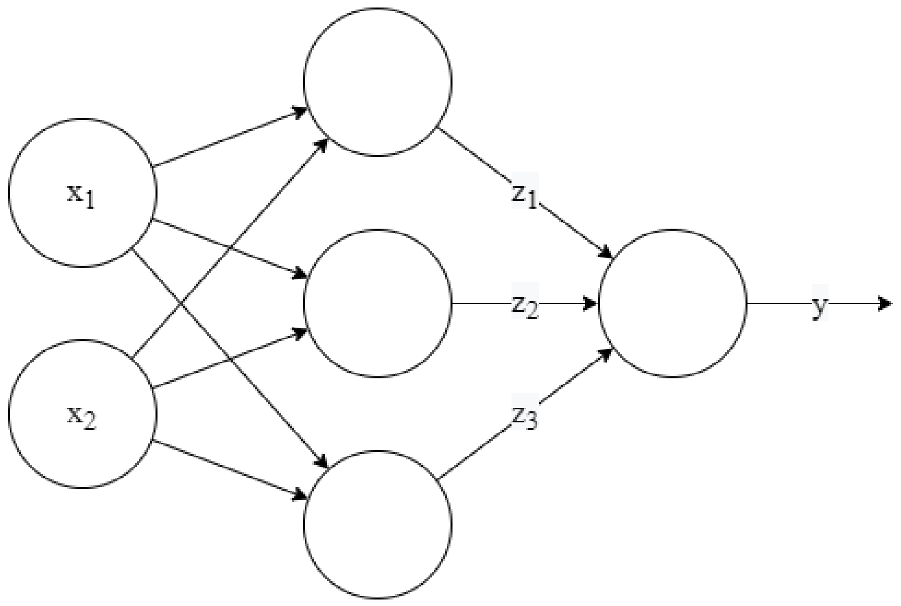


Figure 2.1: **Example of a feed-forward neural network** with two inputs x, one hidden layer with its neurons' outputs: z, and one output that produces y.

neuron of the four in the hidden and the output layers applies its own set of weights and a bias to the input it receives from the preceding layer, for example

$$z_1 = f_z((w_1^{z_1} x_1 + w_2^{z_1} x_2) + b^{z_1}) \tag{2.9}$$

and

$$y = f_y((w_1^y z_1 + w_2^y z_2 + w_2^y z_2) + b^y), \tag{2.10}$$

where and w_i^L are the weights of layer L applied to its i-th input and b^L is the bias of this layer.

The hidden layer from the example above is called fully-connected, owing to the all-to-all connection pattern between its neurons and the outputs of the preceding layer. This example ANN is also feed-forward, as the connections between the layers are strictly one-sided. Another common layer type used in the feed-forward networks is the convolutional layer. Based on the initial attempts at the modelling of

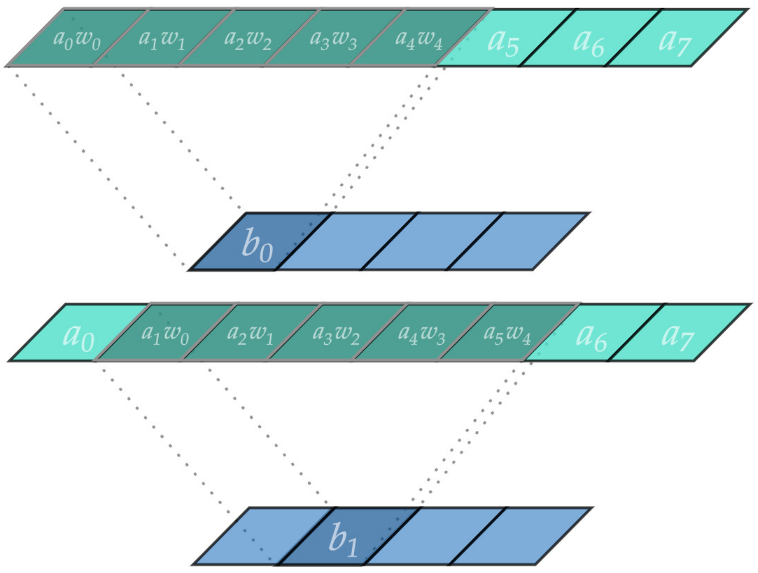


Figure 2.2: **Graphical representation of a single one-dimensional convolutional filter** k applied to obtain the first two elements of the layer output b.

the brain's pattern recognition capability [33], the convolutional layer became the main component of many computer vision systems after its significant successful application for optical character recognition [62]. The convolutional layer iteratively applies a kernel k (also called filter) with some weights $w_i, i \in \{0, .., n\}$ to the subset of its input to produce an element of its output on each iteration. Figure 2.2 presents an example for the first two computations of a convolutional filter of length 5 applied to an input vector a of size 8 with the target output vector b of size 4. In this example, the first element of the output is

$$b_0 = w_0a_0 + w_1a_1 + w_2a_2 + w_3a_3 + w_4a_4 \tag{2.11}$$

and the second one is

$$b_1 = w_0a_1 + w_1a_2 + w_2a_3 + w_3a_4 + w_4a_5. \tag{2.12}$$

The kernel can be said to slide over the input, filtering out some features from each section of the input, with the type of the captured features dependent on the weights w constituting this kernel. The matrix representation of this operation may be found useful for clarification of its mechanism:

$$b = ka, \tag{2.13}$$

where k is the matrix form of the kernel:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} w_0 & w_1 & w_2 & w_3 & w_4 & 0 & 0 & 0 \\ 0 & w_0 & w_1 & w_2 & w_3 & w_4 & 0 & 0 \\ 0 & 0 & w_0 & w_1 & w_2 & w_3 & w_4 & 0 \\ 0 & 0 & 0 & w_0 & w_1 & w_2 & w_3 & w_4 \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \tag{2.14}$$

The convolutional layers in a typical convolutional neural network (CNN) operate on significantly more than a single kernel, thus extracting multiple different features at each layer, that are then further processed by the downstream convolutional layers into increasingly abstract features. For a handwriting recognition system this may mean that the initial layers learn to identify strokes or edges in the inputs, with a subsequent layer capturing the presence of simple shapes formed by these edges, and the final convolutions transforming the detected shape information into an abstract concept that we might call "letter A".

For tasks that require the expansion of some data from its compact representation, an inverse of the convolutional operation was found to enable significant performance gains. The transposed convolutional layers implement this inverse transformation in the ANNs. The underlying mechanism of this layer type is almost exactly the same as the one employed in the convolutional layers. The similarity is evident from both the graphical depiction of this operation (fig. 2.3) and its matrix representation:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} w_4 & 0 & 0 & 0 \\ w_3 & w_4 & 0 & 0 \\ w_2 & w_3 & w_4 & 0 \\ w_1 & w_2 & w_3 & w_4 \\ w_0 & w_1 & w_2 & w_3 \\ 0 & w_0 & w_1 & w_2 \\ 0 & 0 & w_0 & w_1 \\ 0 & 0 & 0 & w_0 \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \tag{2.15}$$

Additional configurations of both the convolutional and the transposed convolutional layers, including padding types, strides and dilation, are presented in the published papers [25], but are not of direct relevance to the current work.

Another ANN layer type widely present in deep learning research is the recurrent layer. Originating from the sequence modelling domain, the core necessity behind its function is the representation of

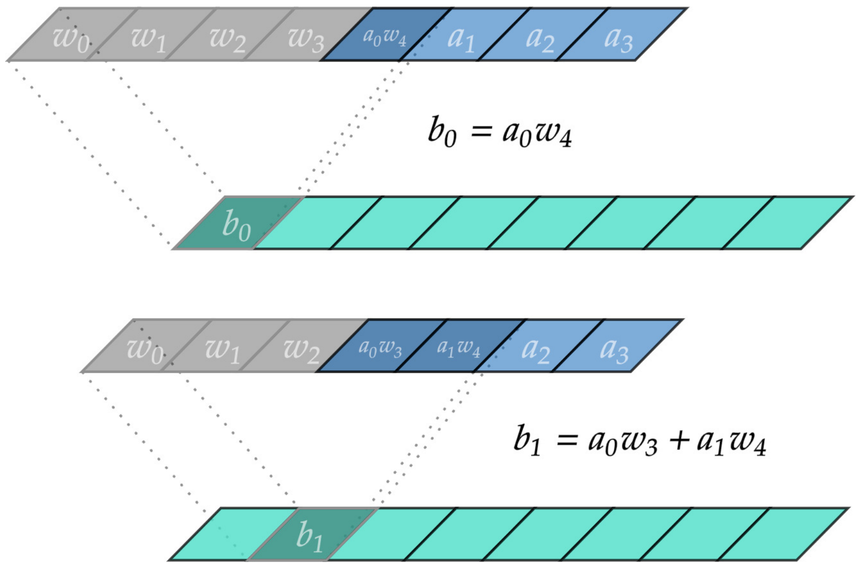


Figure 2.3: Graphical representation of a single one-dimensional transposed convolutional filter k applied to obtain the first two elements of the layer output b.

causality between elements of a, often temporal, sequence, including cases when such a relationship is exerted over long distances in terms of the sequence steps. In its basic implementation, a fully-connected layer described above is enhanced with the recurrent connection, adding its output at step t to the inputs at $t + 1$. Such an implementation did not achieve much success in modelling the target phenomena, but its extension that includes an explicit mechanism for the storage of long-term information proved to be significantly more effective. This recurrent architecture, called the long short-term memory (LSTM) [46], was introduced in 1997, and after almost 20 years of maturation it now occupies an important place in the fields of speech recognition and machine translation.

While mostly applied in their respective historical domains, both the recurrent and the convolutional models are becoming intertwined. Although evidence suggests that convolutional networks may perform comparably or better than recurrent networks on sequence modelling tasks [7], the extension of the convolutional layers with recurrent elements seems promising, and the analysis of the first hybrid convolutional LSTM [100] systems is available in the research literature.

2.2.3 Neural network training

Most of the ANN modules have existed long before the widespread use of deep learning. The key component that enabled the significant advance of their capabilities is the act of training a network, alongside the relatively low price of the modern computational resources that accommodate increasingly large neural networks. The process of training an artificial neural network is similar to the process of shaping a biological neural network, for instance as in the Pavlov's dog experiment. The AI in training is presented with a stimulus, i.e. some input, and is then incentivised in case of a desired reaction and penalised if the resulting behaviour is erroneous. For an ANN this process requires a quantifiable measure of success of the trained network, an objective. A loss function measuring the error of the classification or regression task at hand is a common representation of an objective. As noted in the sections above, the weights and biases that form the connections between the ANN layers determine the transformations that the network inputs are processed with before being output by the ANN. Therefore, the training of a neural network implies the modification of the parameters w, b aimed at the optimisation of the network's performance in terms of the objective function L:

$$\underset{w,b}{\arg\min}\, L(w,b). \tag{2.16}$$

The specific loss function strongly depends on the learned task, as it should accurately quantify the (un)desirable behaviour of the trained network. The standard loss used for classification tasks is the cross-entropy that essentially measures how badly a sample is misclassified. For a predicted probability y_c of the output y belonging to class c, the cross-entropy loss $L_{cross\_entropy}$ is calculated as

$$L_{cross\_entropy}(y,c) = -ln(y_c). \tag{2.17}$$

Thus, in case the sample is correctly attributed to the target class the loss is equal to 0, and it increases towards infinity as the prediction deviates from the correct output probability 100% and towards 0. The standard loss function used in regression tasks is the mean squared error (MSE):

$$MSE(y,\tilde{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2, \tag{2.18}$$

where $y = y_1,..,y_n$ and $\tilde{y} = \tilde{y}_1,..,\tilde{y}_n$ are the vectors with length n of target and predicted outputs respectively. The MSE loss function thus directly measures the average difference between the prediction and the actual value and penalises large errors relatively stronger than smaller ones.

Key for the work described in this thesis is the approach of evaluation of a neural network's performance by another ANN, i.e. the use of a neural network's output as a loss function for another network. This idea forms the core of the method proposed by the authors of the GAN [38], which will be reviewed in the next section. The networks presented in the current thesis models continuous labels and outputs and are thus trained using MSE as the loss function where the adversarial approach is not used.

The concrete mechanism that implements the discrete optimisation steps of the parameters of an ANN is yet another active research topic. The most popular optimisers are the gradient-based methods, like the stochastic gradient descent with momentum [94] and the Adam optimiser [58], built on the foundation of error back-propagation. These algorithms propagate the gradient, which can be thought of as the direction of a slope that leads down towards the optimum of the objective function, from the output error back through the weights and biases used to generate this output. In other words, starting with a small step improvement of the output error, the algorithm calculates the small step changes in the parameters of a preceding neural network layer that would result in the respective output error change. This process is then repeated for each layer, propagating the error change throughout the entirety of the neural network parameter set. The main difference of Adam to the stochastic gradient descent is the adaptive learning rate used for different parameters. This means that the size of the step changes applied to the network's weights during the optimisation adapts to the history of observed gradients by making the step size invariant to the gradient's scale. The formulation of the Adam optimiser update rules that are iteratively applied at each time step are as follows:

$$\begin{aligned}
g_t &= \nabla_\theta L(\theta_{t-1}) \\
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_t &= \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon},
\end{aligned} \tag{2.19}$$

where g_t is the gradient obtained from the loss function, t is the optimisation time step index, θ is the optimised parameter set of the neural network; m_t and v_t are the exponential moving averages of the gradient and of the squared gradient respectively with the exponential decay rates parametrised with β_1 and β_2, which represent the estimates of the mean and the variance of the gradient; $\hat{m}_t$ and $\hat{v}_t$ denote the bias-corrected estimates of m_t and v_t that remove the bias towards zero that the exponential moving averages receive from their initialisation with zero values. The Adam optimiser, as the current de facto standard, is utilised during training of the ANN presented in chapter 3. The model presented in chapter chapter 4 is built based on hyperparameter optimisation, and the choice of the optimiser is included in this process as one of the hyperparameters.

The main drawback of gradient-based methods is the requirement for the differentiability of the loss function and the operations constituting the ANN's layers. While reasonable in many use cases, this requirement cannot be satisfied without additional approximations or simplifications for discontinuous functions. In such cases the non-gradient optimisation methods can be employed, among which the evolutionary optimisation algorithms occupy a significant place in ANN optimisation domain followed by similar methods, like the simulated annealing. Evolutionary optimisation considers the weights and biases or the neural network architecture (or sometimes both) to form a candidate solution to the problem the model is evolved to solve. The optimiser searches the solution space by iteratively evolving a set of candidate solutions using some rules that depend on the specific evolutionary optimisation heuristic. The rules usually define a mechanism for the generation of new candidate solutions based on the existing ones and on some random perturbation and a method for pruning the candidate solution set that bounds the maximal size of the set. While most of such algorithms are explicitly aimed at the search of the objective function optimum, several attempts have been made to implicitly search for the optimum via an open-ended search for novel solutions, e.g. [21]. In these cases, the candidate solutions are formed based on their difference to the existing candidate solutions rather than on their performance on the target task. Proof that such an approach outperforms objective search is limited, but the method has been shown to enhance optimisation process in hybrid systems, especially for problems with complex search space where objective search tends to yield local but not global optimum [117].

For example, interesting solutions to the neural network architecture search problem are available in the published materials [27, 91]. Some evidence suggests that non-gradient methods might be comparable in performance to the gradient-based ones [76, 107], but more substantial evaluation is currently lacking with an overwhelming majority of machine learning research applying the established gradient-based optimisers due to the ease of access to their highly-efficient GPU (graphical processing unit, i.e. the video card) implementations across a wide range of computing software libraries.

Training the ANN is the process of optimisation of the loss functions it is provided with, which usually represent the accuracy of the neural network at modelling a given training set. While this approach may incentivise the neural network to learn the underlying concepts of the modelled phenomena, i.e. to build an "understanding" of it, the ANN may also cheat by memorising only the correct answers without developing interpolation or extrapolation abilities. The resulting neural network shows good performance when evaluated on the data it has been trained on, but fails to produce accurate results on the data it has not yet been exposed to. This generalisation incapability is called model overfitting and may occur with most data-driven methods. A common approach for the identification of overfitting is cross-validation. For this the dataset is separated into the training data and the validation data subsets. The model is then trained using only the training data and evaluated using the validation data. The significantly lower accuracy obtained on the validation data than that on the training dataset would then indicate the presence of overfitting. The standard way of overfitting prevention applicable to any loss-based training of an ANN is thus the early stopping of the training. Under this approach the performance of the neural network on the validation dataset is calculated during training, and the model is trained as long as this performance is improving. Several methods have been proposed that attempt to introduce some form or regularisation into the training process aimed at overfitting reduction. Among the widely used is the dropout method [106] which randomly overrides different neuron's outputs and its gradients with zeros during training. Reduction of co-adaptation between neurons is the intended effect, implying that the network is forced to encapsulate the target processing logic within smaller neural structures by limiting during training the stability of the connections between neurons that are separated by several layers of the neural network. Another common approach that aims to reduce overfitting is weights regularisation. This technique implies the addition of the sum of weights, taken at absolute value in case of L1 regularisation or at squared value in case of L2 regularisation, to the loss function. The model is thus incentivised to follow the Occam's razor principle by making the regularised loss lower for networks using lower values for their parameters all else (i.e., accuracy) being equal. The models described in this thesis are trained with the early stopping policy implemented to prevent overfitting. Empirical tests on the data used in the current work showed that neither dropout nor L1/L2 regularisation improve neither the training time nor the final accuracy of the trained models, thus none of these methods are used with the methods developed in this thesis.

2.3 Advanced ANN architectures

Based on the basic methods introduced above a plethora of large-scale neural network models has been developed to date. The research domain that considers the various architectures and training methods with common uses in the development of AI models is called deep learning. The desire to model complex phenomena requires the used computational framework to support the increasingly intricate non-linear relationships between the analysed or generated concepts. Previous research shows that multi-layer neural networks with many layers, called deep ANNs, provide this desired capability more efficiently than the networks with few but wide layers [10]. As the size of the neural network architectures used in practice, so does the composition of the modules that they are built with. With some of the basic relevant neural network components discussed above, the current section introduces several ANN architectures relevant to the work presented in this thesis.

2.3.1 GAN and Conditional GAN

While the initial success of ANNs has been clear in the discriminative domains, such as the classification of images according to some classes, the research community encountered computational intractability and other difficulties during the application of the same methods for the generative tasks [38]. GAN is a generative neural network that aimed at solving these issues by engaging two networks, the generator G and

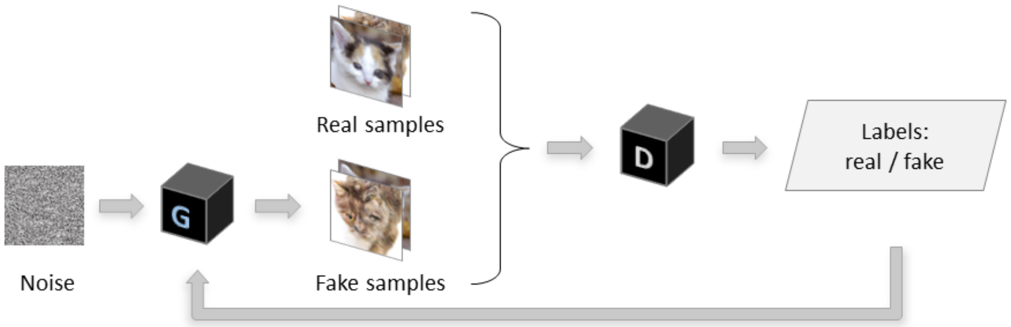


Figure 2.4: **Original GAN architecture.** G denotes the generator network, D - the discriminator.

the discriminator D, in a competitive interaction:

$$\min_{G}\max_{D} \mathop{\mathbb{E}}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathop{\mathbb{E}}_{z\sim p_{z}(z)}[1-\log D(G(z))], \tag{2.20}$$

where x denotes true samples and z is some input that the generator uses to produce fake samples [38], and $\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)]$ denotes the accuracy of discriminator at the identification of x as origination from the true data distribution p_{data}, and $\mathbb{E}_{z\sim p_{z}(z)}[1-\log D(G(z))]$ denotes the probabilities estimated by the discriminator of samples z belonging to the distribution of fake samples. The purpose of the discriminator is thus the identification of whether a given data sample is fake or real, and the generator is trained to produce synthetic data that the discriminator would classify as real. The high level representation of this architecture is depicted on fig. 2.4.

The training of GANs has been accompanied by several problems in the early years since its inception. These include training instability, gradient vanishing due to discriminator overtraining and mode collapse, i.e. the condition of the generator outputting a single output type instead of the many types present in the true data. Over several iterations of research works addressing these issues, the publications converged onto the application of the Earth Mover distance as the loss metric in Wasserstein GAN [6] and its extension with gradient penalty in WGAN-GP [41]. WGAN-GP losses for the generator G and the discriminator D networks are

$$\begin{aligned} L_G^{wgan\text{-}gp} &= -\underset{\tilde{x}\sim\mathbb{P}_g}{\mathbb{E}}[D(\tilde{x})], \\ L_D^{wgan\text{-}gp} &= \underset{\tilde{x}\sim\mathbb{P}_g}{\mathbb{E}}[D(\tilde{x})] - \underset{x\sim\mathbb{P}_r}{\mathbb{E}}[D(x)] + \lambda_{gp} L^{gp} \end{aligned} \tag{2.21}$$

respectively, where $\mathbb{P}_r$ and $\mathbb{P}_g$ are the real and the generated signal distributions, and

$$L^{gp} = \underset{\hat{x}\sim\mathbb{P}_{\hat{x}}}{\mathbb{E}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{2.22}$$

is the gradient penalty, λ_{gp} is its scaling hyperparameter, x and $\tilde{x}$ denote the real and fake signals respectively and $||\nabla_{\hat{x}} D(\hat{x})||_2$ is the square norm of the gradient of $D(\hat{x})$; and $\mathbb{P}_{\hat{x}}$ is a distribution sampled uniformly from straight lines between pairs of points from $\mathbb{P}_r$ and $\mathbb{P}_g$ [41]. Initial empirical tests with GAN training performed in scope of this thesis' work showed that the Wasserstein loss with gradient penalty indeed provides the robustness of training results across multiple runs that was not attainable with the original GAN loss. It must be noted that the calculation of the gradient penalty (eq. 2.22) requires a second call to the discriminator function during gradient evaluation, which increases total training time by approximately 70%. Despite the increased computational load, the stability of training was considered subjectively more important, thus WGAN-GP loss was used during the training of the GANs presented in this thesis.

The conditional GAN is an extension of the original GAN architecture with the additional input that the networks are conditioned on. This is achieved by substituting the random noise input of the generator network in the vanilla GAN for the conditioning labels. In case of a manufacturing simulation model, these labels could be the process parameters that define the configuration of the manufacturing process. A high-level visualisation of the conditional GAN architecture is pre-

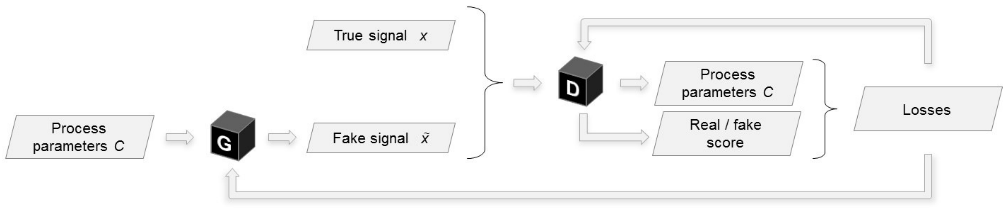


Figure 2.5: **Conditional GAN architecture.** G denotes the generator network, D - the discriminator.

sented on fig. 2.5. The control over the network's output via the input parameters is of interest for the manufacturing process simulation use case, and its application to the proposed GAN model extension will be described in the sections below.

2.3.2 StyleGAN

The digital twin component architecture discussed in the chapters of this thesis is inspired by StyleGAN [56]. It is an image generation model based on two-dimensional deep convolutional networks with the enhancement of the generator by style-injection adopted from style transfer research works. While the traditional GAN feeds the generator inputs directly into the network, in StyleGAN the input is first transformed into an intermediate space via a mapping network. This network transforms the label inputs into style components that are then injected into the layers of the synthesis network via the adaptive instance normalisation (AdaIN) operation [49] defined as

$$\mathrm{AdaIN}(x_f, s_i) = s_i^s \frac{x_f - \mu(x_f)}{\sigma(x_f)} + s_i^b, \tag{2.23}$$

where x_f is a filter response that is each normalised separately and s_i^s and s_i^b are the scaling and the bias components of the style vector at level i. This way AdaIN renormalises the means and standard deviations of each channel of the generator's output at level i with those calculated by the mapping network from the input labels (i.e., the process parameters in our case) and encoded as the style component s_i. The signal generation within the deep synthesis network of the StyleGAN starts with a learned constant vector that is iteratively modulated by the style components introduced at each level, alongside an addition of a noise map input at each layer, and processed by a convolutional layer. The output of each subsequent convolution blocks, each composed of two AdaIN operations and two convolutional layers, is a set of multichannel images with their resolution doubling at each block. Following the StyleGAN architecture, the conditional StyleGAN model proposed in chapter 3 injects the style components into the network via the AdaIN operation. The synthesis is likewise initiated from a learned constant, but the noise inputs are omitted, as the target data model that the generator aims to learn is deterministic, which is discussed further in section 3.2.

2.3.3 CycleGAN

One of the extensions of the GAN architecture for the use in transfer learning is the CycleGAN [131], an image-to-image translation network that utilises mirrored duplex-GAN architecture for image style transfer between two domains. The underlying intent is the training of two generators: one, G_{ab}, that translates data samples from domain a to domain b and the other, G_{ba}, that executes the inverse transformation. The key invention of [131], which necessitates the addition of the second GAN structure, is the cycle consistency loss that enforces the equivalence between the true signals from one of the domains $\tilde{x}$ and the reconstructed signals $\tilde{\tilde{x}}$ that are obtained after passing the true signals through both generators, i.e.

$$\tilde{\tilde{x}}_a = G_{ba}(\tilde{x}_b) = G_{ba}(G_{ab}(x_a)) \tag{2.24}$$

and

$$\tilde{\tilde{x}}_b = G_{ab}(\tilde{x}_a) = G_{ab}(G_{ba}(x_b)), \tag{2.25}$$

where a and b denote the two domains. The cycle consistency loss is calculated for both domains and is defined as:

$$\begin{aligned} L^{cycle} &= |x_a - \tilde{\tilde{x}}_a| + |x_b - \tilde{\tilde{x}}_b| \\ &= |x_a - G_{ba}(G_{ab}(x_a))| + |x_b - G_{ab}(G_{ba}(x_b))|. \end{aligned} \tag{2.26}$$

The use of the cycle loss solves the issue of mode collapse in unpaired image-to-image translation tasks, i.e. the image domain adaptation problem where the training dataset consists of unpaired images from the two domains. For this purpose the L^{cycle} is useful for the domain adaptation of machining signals to the domains with limited data availability, as evidenced by its implementation within the CycleStyleGAN model proposed in chapter 4.

2.4 Chapter summary

The advent of cheap computational power that followed the Moore's law over the last few decades relatively well has made some of the former science fiction technologies accessible today. While not all fantasies have come true to date, and the technological advances have not shed much light on the topic of consciousness, the techniques for AI training and their applications in industrial, social and domestic use cases have challenged the established methods and approaches to many problems. Having identified the research gaps in the previous chapter, the current chapter established the background and the basics underpinning the AI methods proposed in this thesis and introduced the GAN, conditional GAN, StyleGAN and CycleGAN ANN architectures relevant for further discussion. The rest of the work discloses the modelling and analysis results targeted at the manufacturing application of the generative AI in the context of a machining digital twin implementation.

Chapter 3

Conditional StyleGAN modelling and analysis for a machining digital twin

3.1 Introduction

An unexplored area of research of GAN applications in the manufacturing field is the potential use of controllable generative features of a GAN for analysis of the manufacturing processes. Research in this direction can potentially uncover ANN-based data-driven simulation techniques that could significantly augment the decision-making process pipelines in a manufacturing enterprise. This work focusses on simulation via data-driven generation as a component of a future digital twin. GAN is a suitable candidate for digital twin development due to its efficiency at inference time and the generative nature of the model, in addition to the flexibility benefits of a data-driven method that reduce the expected cost of implementation of the model for highly variable processes.

The current work proposes the first StyleGAN-based digital twin machining simulation component. A conditional StyleGAN architecture is developed that captures the conditional distribution of a vibration signal. The process signal generation is controllable via manipulation of the input machining process parameters. The model may thus act as a vibration simulation tool that maps process parameter inputs to vibration signal outputs. This makes the proposed GAN usable as a process optimisation instrument. An optimisation process loop would search for the best process parameters by interrogating the model to obtain parameters-signal pairs and determining the process quality based on the obtained signals. This chapter also introduces a method of uncertainty analysis that is applicable to the optimised process state. To this end, a novel generation sensitivity analysis technique is proposed that aims to estimate the conditions under which the simulation yields reliable results.

The following text describes the data and the neural network model in section 3.2, followed by the methodology and the experimental analysis in sections 3.3 and 3.4 and concluded with a discussion of the implications of the analysis results for potential applications and future research in section 3.5.

3.2 GAN model as a machining digital twin

3.2.1 Dataset: Machining Tool Vibration

Manufacturing process data is a scarce resource at the current moment due to its acquisition cost. While the sensing hardware is getting increasingly cost-effective, the production processes themselves are costly to run, implying that any manufacturing operations yield a sunk cost in case they do not produce a valuable product, which is the case for most experimental operations. And although the data could be collected from the monitoring of a live production process, the variability in its conditions is likely to be insufficient for a data-driven modelling approach to capture the system dynamics across these conditions. Commercial confidentiality of such data adds an additional impediment to its use in public research. It can be expected that ac-

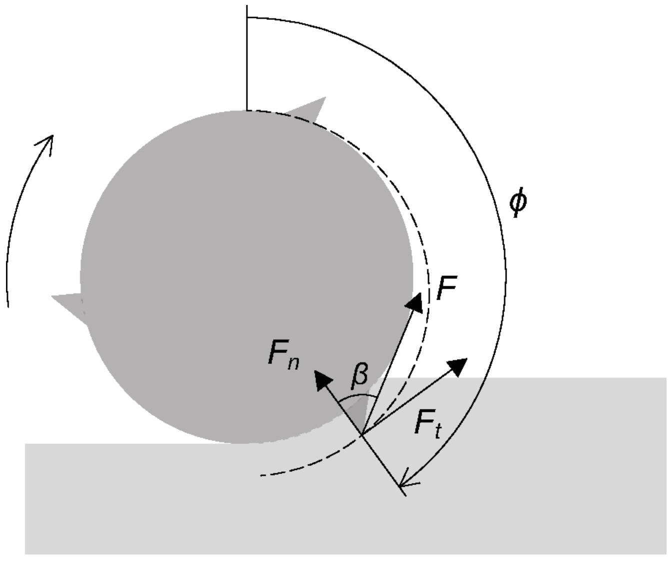


Figure 3.1: Geometrical representation of the forces simulated by the physics-based model. F is the cutting force and β is the force angle. F_t and F_n denote the tangential and the normal cutting forces and ϕ is the cutting tooth angle.

tual implementations of data-driven digital twins would be initially trained on the existing and proven physics-based models, due to the experimental data scarcity mentioned above, and fine-tuned using a mixture of simulated and empirical data. The simulation that produced the dataset used in this work represents a surrogate of a real operational data-generating process. On one hand, this allows for a rigorous analysis of the digital twin component performance due to the full control over data generation. On the other hand, the proposed approach approximates a real world scenario of transition from a pure physics-based modelling to a scenario with mixed physics-based and experimental data.

The GAN model described in this work is trained on a synthetic dataset produced by a physics-based time-domain simulation model adopted from [95]. The simulation iteratively calculates the forces produced by the interaction between the cutting teeth of a non-rigid machining tool and a rigid workpiece (Figure 3.1). These forces are used

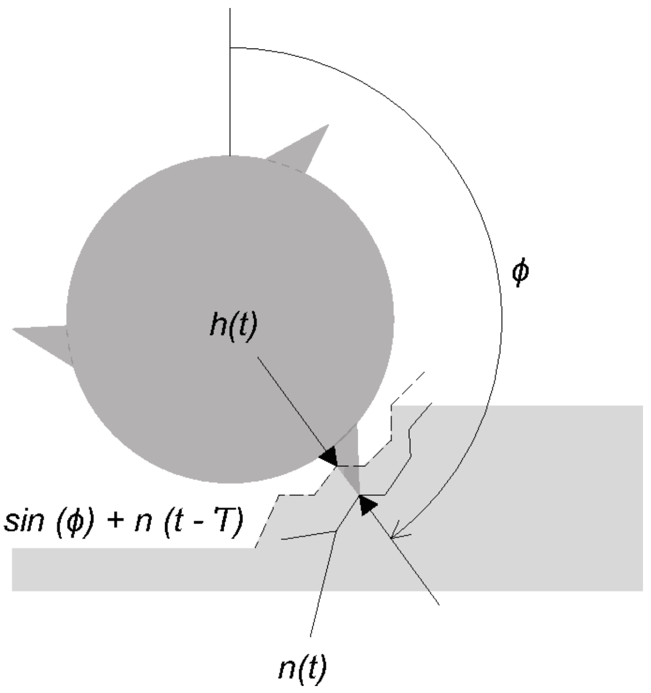


Figure 3.2: **Workpiece geometry produced by the simulation model.** $h(t)$ is the instantaneous depth of cut at time t that is the distance between the current normal direction vibration level $n(t)$ at angle ϕ and the cut surface at angle ϕ produced at time $t - T$, where T is the time period of cutting tool revolution between two neighbouring cutting teeth.

Table 3.1: Milling time-domain simulation parameters

Parameter type	**Parameter**	**Value**
Machining parameters	chip width b	0.004 to 0.005
	spindle speed ω	3000 to 4000
	feed rate f	10.2
Process-dependent parameters	number of cutting teeth N_t	3
	start angle of cut ϕ_s	126.9
	exit angle of cut ϕ_e	180
	process dependent coefficient K_s	2250e6
	force angle β	75
	x direction dynamics parameter k_x	9e6
	x direction dynamics parameter ζ_x	0.02
	y direction dynamics parameter k_y	1e7
	y direction dynamics parameter ζ_y	0.01
Simulation parameters	steps per revolution	256

Algorithm 3.1 Physics-based simulation algorithm

```
1: for $t$ in range(simulation time steps) do
2:     $h(t) \leftarrow$ Instantaneous chip thickness
3:     $F(h) \leftarrow$ Cutting force
4:     $Vibration(F) \leftarrow$ Displacement values
5:     $\phi \leftarrow \phi + d\phi$ (increment tooth angle)
6: end for
```

in the derivation of the acceleration, velocity and displacement, i.e. the vibration, of the cutting tool. Vibration is selected as the analysed signal type based on the low expected cost of its acquisition and potential usefulness in the analysis of the machining process. The simulation tracks the position of each cutting tooth and the workpiece geometry produced by material removal (Figure 3.2) to identify which cutting teeth are performing the cut at each time step. The operation considered in the current work is a linear non-slotting milling cut performed with a straight-teeth cutting tool on a metal workpiece.

The physics-based model accepts several variables that control the deterministic simulation, including the machining parameters controllable during the configuration of the metal cutting process and the parameters dependent on the characteristics of the workpiece material, the machining tool and the manufactured product. These variables are detailed in Table 3.1 followed by the values used for generation of the training data. The parameter values are based on the example from [95] that describes a linear down milling cut performed by a three-tooth square endmill with straight teeth on a low-carbon steel alloy, the type of cut pictured on Figures 3.1 and 3.2. The parameter values are constant throughout each cutting operation, and the parameters varied across the samples in the produced dataset are chip width and spindle speed in ranges from 0.004 to 0.005 mm and 3000 to 4000 rpm respectively. The generated signals represent the displacement of the cutting tool along the x-direction during the third revolution of the cutting tool, sampled at a rate proportional to the spindle speed.

A signal sample is obtained for each combination of 200 linearly spaced chip width and 200 spindle speed parameter values in the specified ranges, resulting in 40 000 signal samples within the dataset. The only pre-processing applied to this data is the mean and standard deviation normalisation that is applied to each of the process parameters separately and to the time-domain signals. The validation dataset containing 40 000 samples is produced using the same approach, but with the process parameters values shifted a half of the step, i.e. chip width from 0.004025 to 0.005025 and spindle speed 3002.5 to 4002.5. Both training and validation datasets are publicly available at `https://gitlab.com/EZotoff/conditional-stylegan-digital-twin`.

3.2.2 Conditional StyleGAN architecture

Elements of StyleGAN introduced in section 2.3 are repurposed for the 1D case of a time-domain signal simulation developed in this chapter. The noise inputs and the mixing regularisation (regularisation applied during training that randomly mixes the disentangled latent with another one to produce a sample from the generator G) are excluded from the model, as the variation of outputs of the target distribution is deterministic with respect to the input process parameters, i.e. the training data contains not more than a single sample for each unique label set. The architecture is enhanced with the substitution of the random input latent vector for continuous labels C: the machining process parameters, chip width and spindle speed, that the generator's output is conditioned on. The process parameters are used as inputs to the generator and as outputs of the discriminator, i.e. the discriminator learns to not only identify synthetic data samples, but also to estimate the labels associated with a given time-series. The architecture includes a non-linear mapping network M that projects process parameter inputs into a disentangled latent space. The styles S produced from the input labels C by the mapping network M subsequently control the modulation of outputs of the convolutional layers within the synthesis

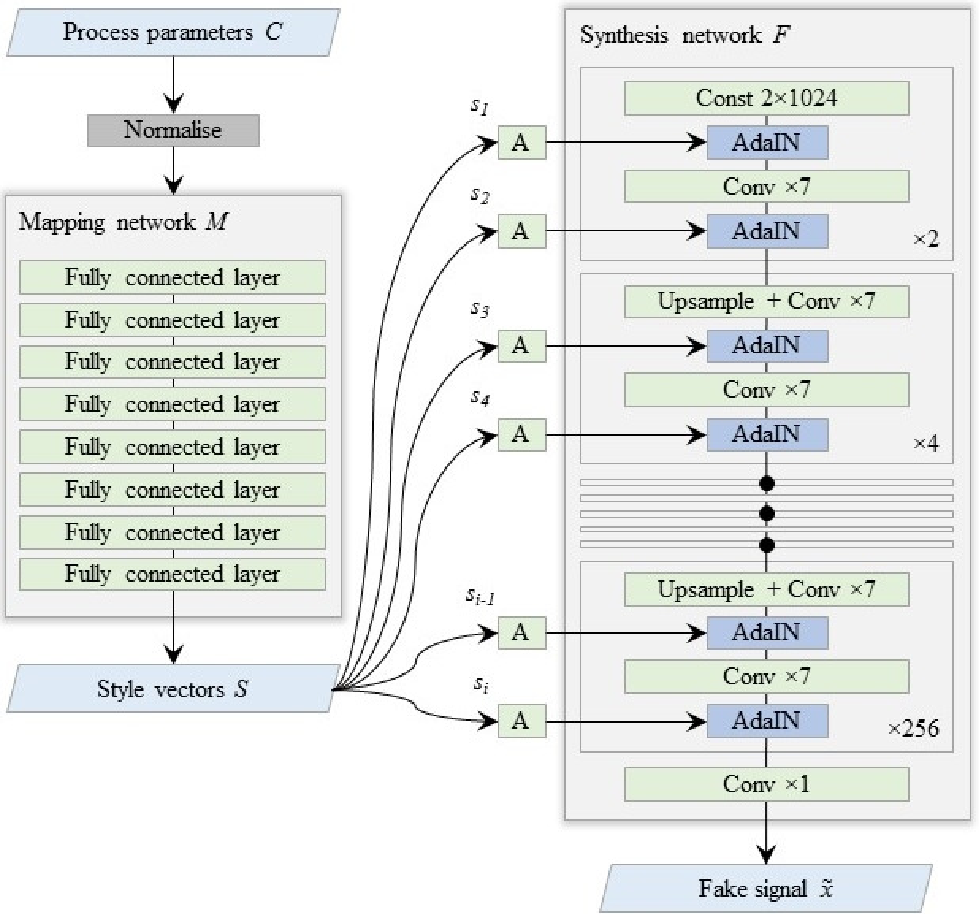


Figure 3.3: **Architecture of the conditional StyleGAN generator network.** "A" denotes learned affine transformations of style components s_i; "AdaIN" - adaptive instance normalisation [49], which implements the per-channel modulation of the network's intermediate outputs by the transformed style components.

network F of the generator (see Figure 3.3).

The mapping network M is implemented as a multi-layer perceptron and consists of 8 layers with 32 neurons each with leaky ReLU activation functions. M maps the input process parameters C into style vectors S of length 256. The first input to the synthesis network is a learned constant vector of dimension 2x1024, where the first dimension is the signal length and the second dimension is the number of filters. This learned constant is sequentially processed by multiple blocks each containing two convolutional layers with convolutional kernel of size 7. Except for the first block where the constant vector takes place of the first convolutional layer's output, the first convolutional layer in each block upscales the signal length by a factor of 2 and reduces the number of filters by a factor of 2 until the number of filters reaches 64. After each convolutional layer the signals are passed through a leaky ReLU activation function and then normalised and mixed with the respective style component vector within the AdaIN operation. The last layer of the synthesis network F applies a convolution operation with kernel 1 to aggregate the 64 filter outputs produced by the final block into a single signal of length 256.

The loss functions are adjusted to accommodate the inclusion of machining process parameters in the network's architecture by addition of terms that penalise inaccurate label predictions. This is similar to the approach followed by the authors of InfoGAN [16], with the following difference. The accuracy of label predictions for training data L_D^{info} impacts only the discriminator, while the accuracy of the predictions for fake data samples L_G^{info} is taken into account only by the generator. The loss terms are

$$\begin{aligned} L_G^{info} &= \sqrt{\frac{1}{n}\sum_{j=1}^{n}(c_{t,j} - \tilde{c}_{f,j})^2}, \\ L_D^{info} &= \sqrt{\frac{1}{n}\sum_{j=1}^{n}(c_{t,j} - \tilde{c}_{r,j})^2}, \end{aligned} \tag{3.1}$$

where $\tilde{c}_{f,j}$ is a value of parameter j predicted by the discriminator based on a fake signal, $\tilde{c}_{r,j}$ is a value predicted from a real signal, and $c_{t,j}$ are the true parameter values. On one hand, the generator is thus incentivised to encode the label information in an identifiable way within the synthesised samples. On the other hand, the discriminator learns the relationship between labels and samples only on the real data, thus preserving the non-cooperative nature of the minimax game between the generator and the discriminator. With λ_{info} denoting the scaling factor for the label prediction accuracy error loss, the total loss functions for the generator L_G and the discriminator L_D are therefore:

$$\begin{aligned} L_G &= L_G^{wgan\text{-}gp} + \lambda_{info} L_G^{info}, \\ L_D &= L_D^{wgan\text{-}gp} + \lambda_{info} L_D^{info}, \end{aligned} \tag{3.2}$$

with $L_G^{wgan\text{-}gp}$ and $L_D^{wgan\text{-}gp}$ are the Wasserstein losses with gradient penalty defined in equation 2.21.

The generator network learns the conditional distribution of the time-domain vibration signal with respect to the machining process parameters that control the basic milling conditions. The use of style-based architecture of the generator enables a reduction of a trained model's complexity via inspection of its style-level components. This and the degree of control provided by the conditional component of the generator increases the model's potential interpretability and enables a reductionist approach to the analysis of its black-box inner mechanisms.

The models are trained using the Adam optimiser for both the generator and the discriminator with learning rates of 0.00001 and 0.0001 respectively. The network losses are parametrised with $\lambda_{info} = 10$ and $\lambda_{gp} = 10$. The GAN described in this work is trained until convergence, and the models used in correlation analysis described in Section 3.4.1 were trained for the same number of epochs as the main network, and thus have not necessarily converged. Training convergence was assessed based on the rate of improvement of the root-mean-square error (RMSE) metric measured and averaged across the validation dataset.

3.3 Digital twin performance analysis

The generator successfully learns to capture the relationship between the process parameters and the time-domain signal and performs well

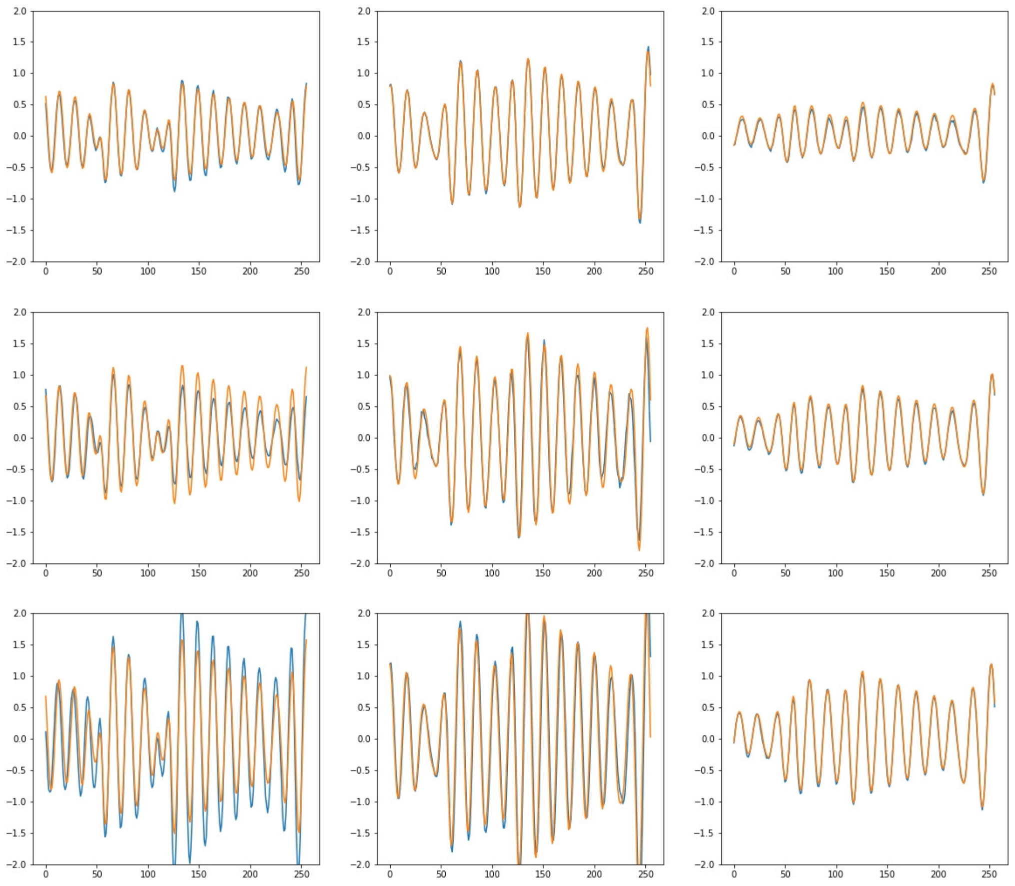


Figure 3.4: **Comparison of generated time-series samples and validation data samples.** X-axis represents time steps, Y-axis - displacement in log scale. Real signal is represented by the yellow curves and the generated signal by blue.

both on training and validation data. Figure 3.4 depicts several samples of generated time-series against the signals from training data produced using the same process parameters.

The metrics used in the analysis of the generator network performance are aimed at capturing the accuracy of the model conditional on the input parameters. The potential mode collapse (i.e. the inability of the generator to produce parts of the target distribution) inevitably affects the accuracy of the generator due to the deterministic nature of the experimental data coupled with the conditional generation. There-

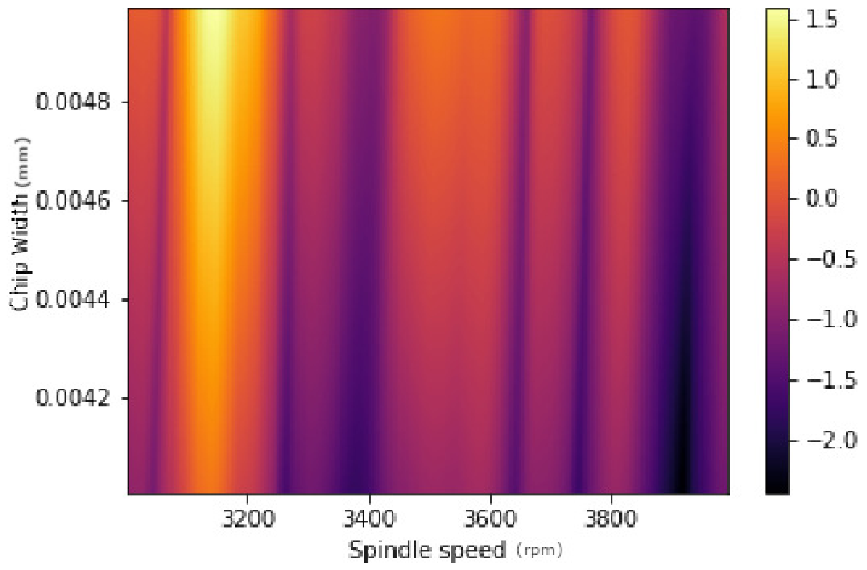


Figure 3.5: **Standard deviation of training data samples** for each set of process parameters values (log scale).

fore, the discussed experimental setting permits less attention on the variety of the generated samples, and the current work focusses on the analysis of the generator accuracy. The true accuracy value for some labels C is measured by the root-mean-square error (RMSE) $\mathcal{E}(C)$ between the true signal $x(C)$ from the training data and the synthetic signal $\tilde{x}$ produced by the generator:

$$\mathcal{E}(C) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i(C) - \tilde{x}_i(C))^2}, \tag{3.3}$$

where n is the signal length.

The generative performance of the neural network is investigated via analysis of metrics mapped across machining process parameter values, chip width and spindle speed. This is visualised by calculating the inspected metric for a range of the process parameter pairs and plotting the values on a two-dimensional figure with spindle speed varied across the horizontal axis and chip width across the vertical axis. The training data exhibits high variability, especially across the spindle

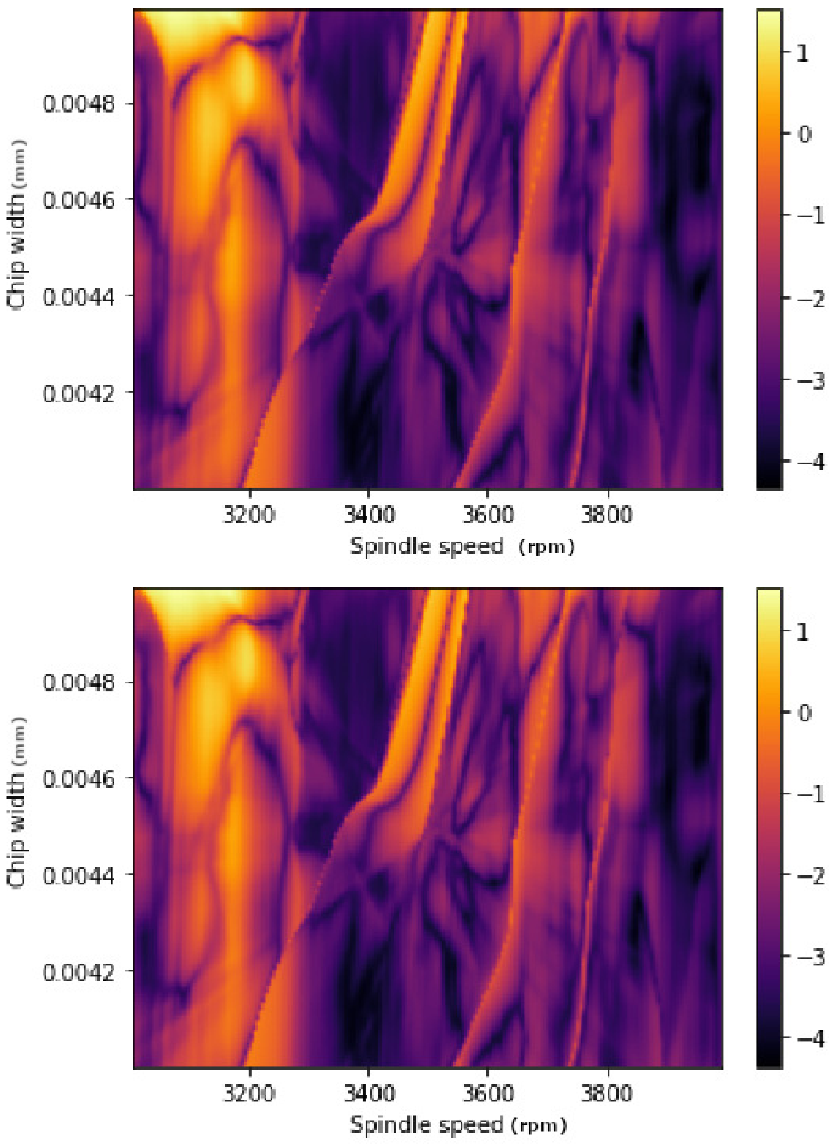


Figure 3.6: **Error $\mathcal{E}$ values** in log scale for generated time-series across various process parameter values for training data (top) and validation data (bottom). A single point on the map represents the generator $G(C)$ error value $\mathcal{E}$ measured at process parameter values of chip width and spindle speed C corresponding to the Y-axis and the X-axis values respectively. The error pattern similarity on both maps indicates low over-fitting.

speed parameter values range, visible on the plot of standard deviations of the training data signals (Figure 3.5). Figure 3.6 depicts the error $\mathcal{E}(C)$ of the generated time-series on training and on validation data. The mean (standard deviation) of these errors are approximately 1.408 (2.554) and 1.408 (2.514) millimetres respectively. Error distributions are nearly identical, i.e. the generator performs equivalently during both training and validation. The validation dataset is of the same size as the training dataset, which is feasible due to the control over the source data synthesis.

In contrast to the high accuracy of the generator in the areas of the parameter space characterised by low dispersion in the training data, the generative performance is suboptimal in some regions of high training data variance. Closer inspection of a region between 3450 and 3550 spindle speed reveals that the generator experiences a local mode collapse at high chip widths (for an example refer to Figure 3.7). The trough visible on the error map in this region represent a parameter space where the generator successfully learnt the mode of the target signal, while the peaks to the sides along the X-axis from this narrow band of high accuracy are indicative of the dropped modes. An inspection of the dynamics of change of the training data signals compared to the change of the synthesised signals reveals that whereas the training data signals change shape linearly with the change in spindle speed, the generated signals remain constant for most of the inspected parameter space and sharply switch to the next mode near the peak on the error map.

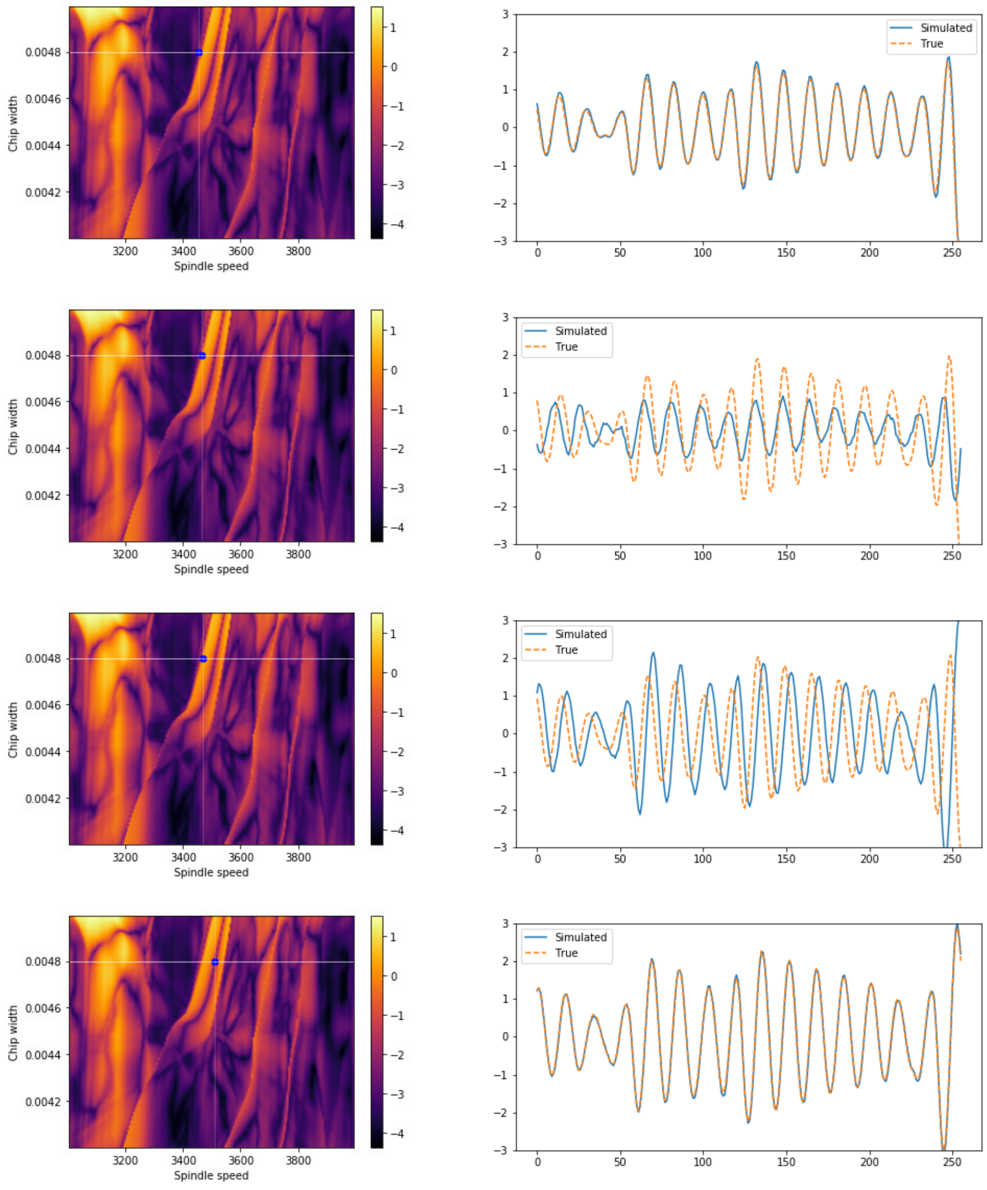


Figure 3.7: Example of a path in labels space where the transition between the signal modes is smooth in the training data, but abrupt in the signals produced by the generator. The blue dot on the error maps (left) indicates the parameter values used to compare the two signals (right), real signal in yellow and generated signal in blue. The differences between the fake signals along the labels transition paths shown on the two top figures and the two bottom figures are much lower than for the real signal. The opposite is true for the transition captured by the two middle figures.

3.4 Digital twin sensitivity analysis

3.4.1 Generator sensitivity

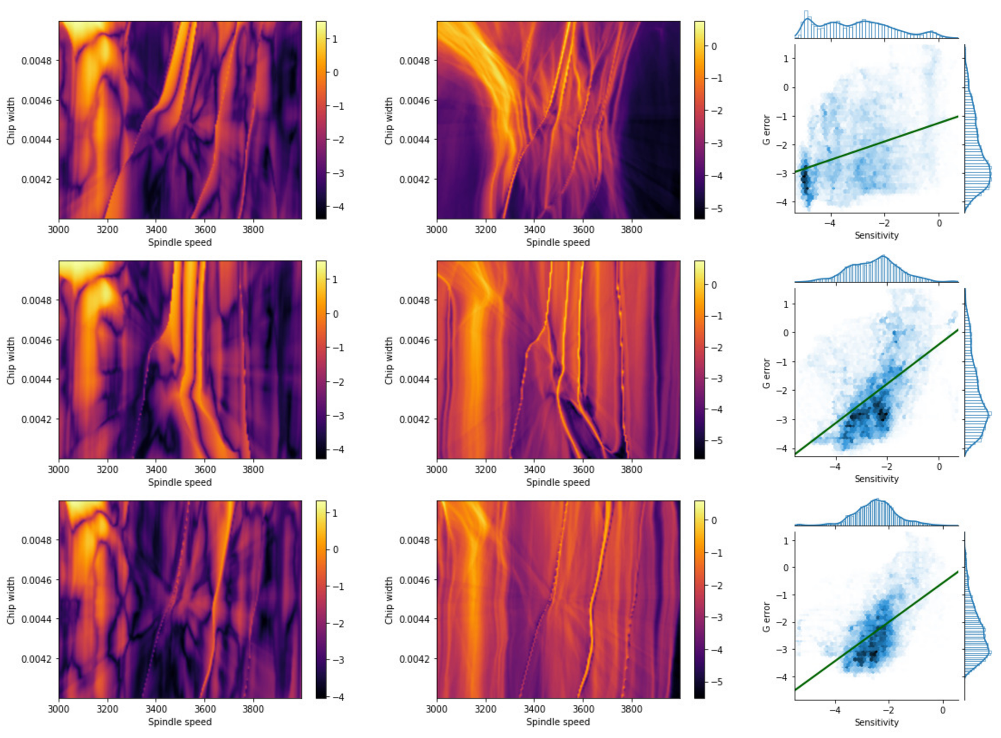
The metric discussed in the following sections is the sensitivity of the generator output to the input parameters δ:

$$\delta(C) = \frac{1}{m} \sum_{j=1}^{m} \left(\left| \frac{\partial \tilde{x}(C)}{\partial c_j} \right| \right), \tag{3.4}$$

where $c_j \in C$ and m is the number of input parameters in a label set C.

The generator sensitivity $\delta(C)$ provides an insight into the origin of the parameter space regions where the GAN produces samples with relatively high inaccuracy. Empirical observation of the sensitivity maps obtained from several trained GANs reveals a complex but stably recurring relationship between the distributions of the generator sensitivity $\delta(C)$ and the generator error $\mathcal{E}(C)$ across the parameter space. Comparison of the correlation between the point-wise sensitivity and error metrics for different generator networks implies a moderately strong connection between these metrics that is nevertheless consistent across the generators, with average correlation of 0.45 for 52 different generator networks. Examples of this comparison are presented on Figure 3.8. Both the empirical observations and the correlation analysis imply that a complex, potentially non-linear, relationship between the two metrics exists. If mapped onto the generator accuracy, the generator sensitivity $\delta(C)$ can be used as an accuracy estimator. Thus, the sensitivity can represent the generation uncertainty at some input parameter values using only the information obtained from the generator model itself. Calculation of $\delta(C)$ requires generation of an additional sample from the generator per parameter in C for each evaluated signal, which yields a negligible additional computational cost for GANs implemented in most modern neural network libraries that allow batch evaluation of the models. Generator sensitivity can thus be efficiently used at inference time to enhance the produced data samples with an accuracy uncertainty metric without requiring any external data and additional measurements.

The discussed relationship between the generative accuracy and sensitivity to input labels implies that the cause of the modes of failure of the generator originates from the parameters inputs and consequently from their style representation. This link is studied in the following section.



(a) Error $\mathcal{E}$ (b) Sensitivity δ (c) $\mathcal{E}$ and δ

Figure 3.8: Error $\mathcal{E}$ and sensitivity δ maps for three trained models, a model's plots on each row. The right column contains plots of the joint distributions of these two measures, with the shaded regions representing the frequency of samples for the corresponding $\mathcal{E}$ and δ values and the solid lines are the fitted linear regressions.

3.4.2 Interpolation Analysis

If the origin of the high sensitivity (which is linked to the high error) can be found, an approach for dampening the sensitivity could be applied, theoretically trading reduced maximal model error for increased average error. The style-based neural network architecture enables a reduction of the model analysis complexity via the inspection of the influence of the disentangled input parameter vectors s_i at the different layers i of the synthesis network $F(S)$ within the GAN, where $S = \{s_i\} = M(C)$ is a set of disentangled style vectors produced by the mapping network M of the generator from the input label vectors C. This is performed via the analysis of the changes in the generated output signals arising from alteration of the disentangled inputs. Two style sets are obtained from the mapping network using different input parameters,

$$S^o = M(C^o) \text{ and } S^t = M(C^t) \text{ for } C^o \neq C^t, \tag{3.5}$$

and two signals are generated using these parameter sets,

$$\tilde{x}^o = F(S^o) \text{ and } \tilde{x}^t = F(S^t). \tag{3.6}$$

An interpolated signal $\tilde{x}^c$ is calculated by feeding into the synthesis network an affine combination of style sets S^o and S^t:

$$\begin{aligned} \tilde{x}^c &= F(S^c), \\ S^c &= \{s_i^c\} = \{s_i^o * w_i + s_i^t * (1 - w_i)\}, \end{aligned} \tag{3.7}$$

where i is the index denoting the style level increasing towards the output layer of the model and w_i is the relative weight of the first style used at the i-th level. In other words, at each style level i the style component of S^c is the weighted sum of the components of S^o and S^t at this level, with the weight $w_i = 0$ indicating that only the component of the first style set S^o is used and $w_i = 1$ implying that only the second

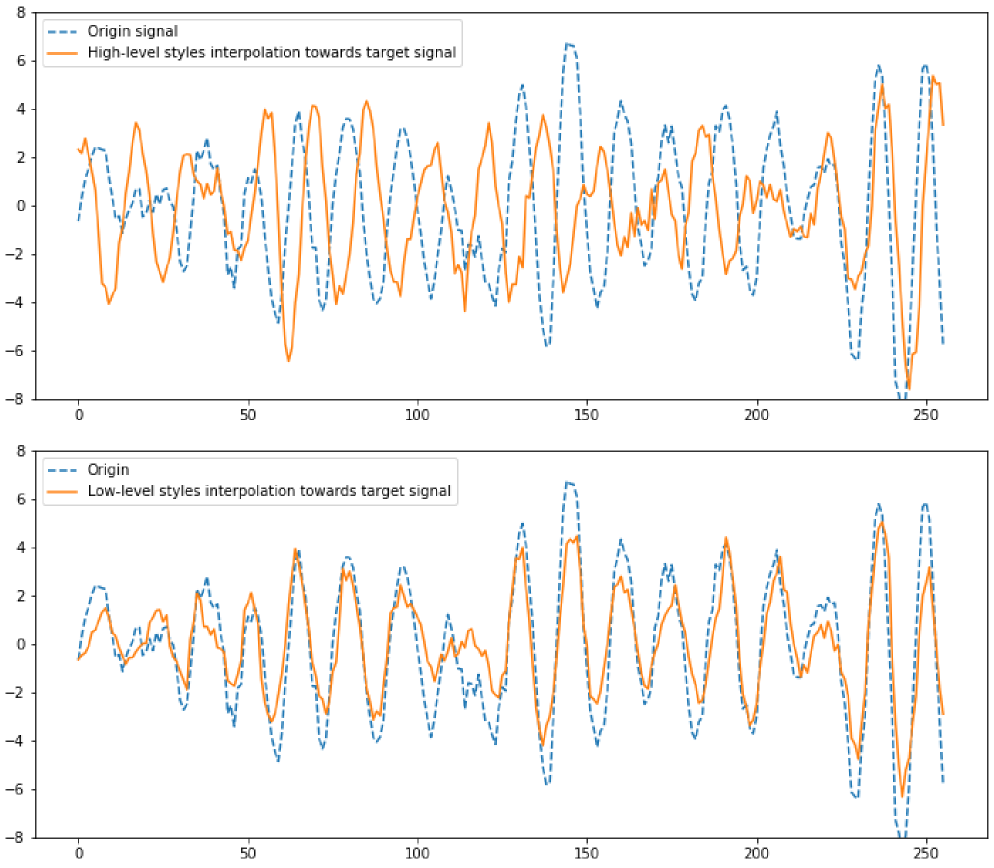


Figure 3.9: **Comparison of the interpolations over high- and low-level styles** on the top and the bottom figures respectively. The dashed line represents the initial signal $\tilde{x}^o = F(S^o)$ on both plots. Top solid line, the signal generated using interpolated styles $S^c = \{s^t_{1..5}\} \cup \{s^o_{6..16}\}$, is significantly different from $\tilde{x}^o$ in its phase and modes. Bottom solid line, the signal produced from $S^c = \{s^o_{1..5}\} \cup \{s^t_{6..16}\}$, differs from the source in its local high-frequency features.

style S^t is applied at this layer.

The variation of the generated signals produced as a result of gradual changes to one or several style components reveals the features that the style components at the respective levels control. By performing a linear interpolation between s^o_i and s^t_i for each $i = k$ individually, i.e. while keeping s^c_i and s^o_i equal for each $i \neq k$, we empirically observe that the generator model style layers can be classified into two groups: high-level styles S_H and low-level styles S_L, where $S_H = \{s_{1..5}\}$ and $S_L = \{s_{6..16}\}$. The high-level styles significantly affect the low-frequency features of the output signal like its phase and general envelope shape. The low-level styles impact the high-frequency detail of the generated output. Figure 3.9 visualises the end points of this interpolation: an initial signal $\tilde{x}^o$, its interpolation towards $\tilde{x}^t$ in high-level styles only (i.e. $\tilde{x}^c = M(S_H^t \cup S_L^o)$) on the top plot and in low-level styles only (i.e. $\tilde{x}^c = M(S_H^o \cup S_L^t)$) on the bottom one.

An analogous inspection of the style-level interpolation effects is performed at the layer level of the generator, i.e. the variation of the activations produced by the intermediary convolutional layers of the model is observed as a result of changes in the style parameters. Empirical analysis reveals that the sensitivity of the layer activations to the changes in the high-level style components exhibits the irregular non-smoothness similar to the sharp gradients notable in $\delta(C)$. The search for the origin of the error-producing high sensitivity discussed in section 3.4.1 can thus be narrowed to the layers of the GAN where the high-level styles are injected. The impact of the style modulation measured at a single-layer level seems insignificant, as evidenced by the minor difference between the distributions of the input and output activations of the style injection layers. Example of this is depicted on Figure 3.10. Nevertheless, these minor changes are propagated through the generator network and magnified by the complex black box processing within the downstream convolutional layers, resulting in major variability of the network outputs.

3.4.3 Activation node sensitivity

One of the possible ways of further reduction of the complexity of the analysed chaotic system is an additional refinement of the analysed object's scale. Thus, the sensitivity of individual activation nodes within the generator layers relative to the sensitivities of other nodes in the same layer is of interest. This sensitivity δ_{node}^{layer} is measured as

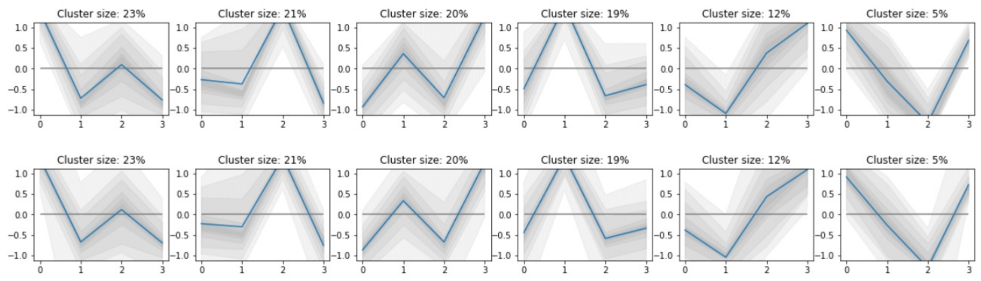


Figure 3.10: **Distributions of activation clusters** that are the input (top plot) and the output (bottom plot) of a high-level style modulation layer of the generator network. Solid line is the cluster activations' mean and the shaded areas show the cluster activations' distribution covering the whole range of activation values in a cluster. Style injection at this layer changes neither the cluster sizes nor the activation distribution significantly.

the sensitivity of the activation outputs to the changes in input process parameters averaged across a set of process parameter values $\mathcal{C}$.

$$\begin{aligned} \delta^{layer}_{node} &= \left| \frac{1}{\sigma k} \sum_{p=1}^{k} (\delta'(C_p)) \right|, \text{ where} \\ \delta'(C_p) &= \frac{1}{m} \sum_{j=1}^{m} (\frac{\partial z^{layer}_{node}(C_p)}{\partial c_j}), \end{aligned} \tag{3.8}$$

$z^{layer}_{node}(C_p)$ is the activation output of a single node within a layer, δ' is the sensitivity of this activation to change in the process parameter input measured at process parameter values C_p and σ is the standard deviation of δ' across the process parameter values.

Therefore, δ^{layer}_{node} is high for the activation nodes that are highly sensitive to the process parameters C robustly across the parameter space. Comparison of δ^{layer}_{node} calculated using the complete process parameter set from training data with δ^{layer}_{node} based on a limited parameter set containing only the parameter values that produce the top 5% of generator sensitivity $\delta(C)$ is presented on Figure 3.11. Most of the nodes producing high sensitivity in regions where the whole generator network is

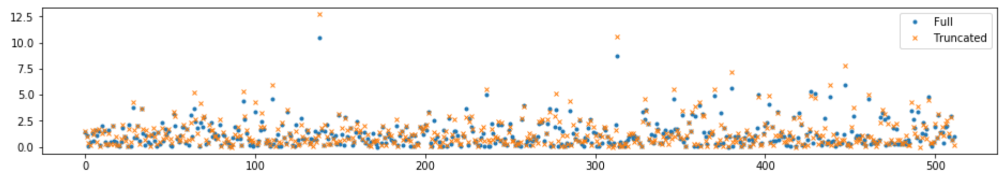


Figure 3.11: Distribution of the activation node sensitivity metric δ_{node}^{layer} for one of the generator layers with values of δ_{node}^{layer} along the Y-axis and node indices along the X-axis. Crosses denote the sensitivity values measured based on the process parameters values from the regions where the whole generator network is highly sensitive to input. The δ_{node}^{layer} calculated on the full process parameter set are marked by the filled dots.

highly sensitive to input are also producing relatively high sensitivity in the process parameter regions not characterised by high generator sensitivity. This metric thus makes possible the identification of the most sensitive activation nodes within the network and consequently the convolutional kernels that produce this sensitivity, which can potentially be used for remedying the anomalously high sensitivity $\delta(C)$ of the generator and reducing the errors in the respective regions.

3.5 Chapter summary

The conditional neural network architecture described in this chapter allows a significant degree of control over the generator via controllable input process parameters, thus enhancing the flexibility of the model and creating the potential for exploratory analysis of the modelled process. Such analysis could serve the purpose of CNC machining process planning and optimisation. Researchers have shown how signal data can be utilised to predict product conformity to quality standards [44, 85], therefore enabling the prediction of manufacturing errors and product quality prior to manufacturing when coupled with signal simulation proposed here. Additionally, the machining process stability estimation can be enhanced with a generative model substitution for some of the physical measurements, which is a future research direction also proposed by machining stability experts [39].

The validation of the proposed GAN model using an almost raw vibration signal data, pre-processed only by domain-agnostic mean and standard deviation normalisation, implies that the findings discussed in this work are generalisable and relevant for other time-domain signal generation applications. The neural network architecture at the base of the proposed model is computationally cheap at inference time. This and the generative nature of GAN enable the development of a machining digital twin component that simulates the underlying physical process in real-time, which is an important step towards implementation of data-driven simulation models in the development of digital twins for Industry 4.0. The proposed analysis methodology utilising the style-based input disentanglement enables a reductionist approach to the model's performance analysis, and the generative uncertainty metrics based on the generator sensitivity presented in this work increase the model's transparency and interpretability. Both are important barriers to the widespread adoption of complex models in the industrial context [50].

The conditional component of the proposed GAN is naturally extensible to simulation of longer signals via autoregressive model based generation, i.e. the inclusion of signal history as a conditioning input to the generator. The use of recurrent neural network layers might also prove beneficial for this extension of the generator network. Conditioning can additionally be applied for adoption of the proposed model to a signal transformation problem, e.g. for prediction of cutting forces from the vibration signals, similar to the image style transfer idea [49].

A real world implementation would likely be limited in terms of data availability due to the relatively high cost of acquisition of non-production experimental data. With the lack of data for creation of a complete map across the various operating conditions, the approach would have to be implemented using a limited subset of these conditions. An attentive consideration of the sampling efficiency of true data would thus be important in the optimisation of the model's ability to learn with fewer data. This could also be remedied by a hybrid data generation approach that assembles the training and validation datasets using any existing expert- or physics-based models complemented with the real world data.

The problem of data scarcity in the context of machine fleet deployment is the topic of the next chapter, which builds on the modelling approach presented above to propose a solution that leverages the knowledge already embedded in an existing process model in the modelling of a process from another domain.

Chapter 4

CycleStyleGAN-based knowledge transfer for a machining digital twin

4.1 Introduction

The widespread digitalisation within the transition towards Industry 4.0 creates a drive for flexible and efficient data-driven simulation modelling. The existing simulation solutions frequently lack the required flexibility and integration access to be effective in a heterogeneous and dynamic environment of the interconnected cyber-physical systems. Nevertheless, the knowledge contained within these costly models is often valued at a very high price. The efficient use of this knowledge is therefore an important concern for any business employing such simulations.

The work presented here proposes a solution for extraction of the knowledge from the existing manufacturing simulation tools applicable both to physics-based and to data-driven source models, thus enabling business cost optimisation. The proposed approach implements a novel CycleStyleGAN domain adaptation model by introducing the style-based signal representation into the CycleGAN framework. The sections below present an evaluation of the effectiveness of the proposed knowledge transfer method and its comparison to an incremental learning approach validated under identical conditions. A proposed use case of the developed model for manufacturing process optimisation is also discussed in section 4.4.

4.2 Materials and Methods

4.2.1 Milling Vibration Datasets

The physics-based model introduced in section 3.2.1 accepts several variables that influence the deterministic simulation, including machining parameters that can be controlled during the metal cutting process configuration and parameters that are dependent on the workpiece material, machining tool, and the manufactured product characteristics. Three datasets are created for the experiments presented in this chapter. Dataset 1 represents source domain data. Datasets 2 and 3 correspond to the target domains that differ from the source domain, respectively, either slightly or significantly. The similarity between dataset 2 and dataset 1 portrays a situation of a small difference in the environment temperature or the machined material properties between the two domains. Dataset 3 represents a case of substantially varied properties of the underlying signals, for example resulting from a change of machining tool. The parameter changes made to obtain the two target-domain datasets were chosen based on an empirical comparison of the resulting datasets to the original dataset with the purpose of introducing relatively small and large changes to dataset 2 and 3 respectively, as described above. While the resulting parameter values are not directly inferred from a particular real combination of tool and material properties, they lie within the ranges used in the examples in [95], and are thus considered realistic.

Table 4.1 details the process simulation variables, followed by the values used to generate the training data. The hyphens indicate the

Table 4.1: Milling time-domain simulation parameters

Parameter type	**Parameter**	**Dataset 1**	**Dataset 2**	**Dataset 3**
Machining parameters	chip width b	0.004 to 0.005	-	-
	spindle speed ω	3000 to 4000	-	-
	feed rate f	10.2	-	-
Process-dependent parameters	number of cutting teeth N_t	3	-	-
	start angle of cut ϕ_s	126.9	-	-
	exit angle of cut ϕ_e	180	-	-
	process dependent coefficient K_s	2250e6	1950e6	1950e6
	force angle β	75	-	-
	x direction dynamics parameter k_x	9e6	-	7e6
	x direction dynamics parameter ζ_x	0.02	-	-
	y direction dynamics parameter k_y	1e7	-	1.3e7
	y direction dynamics parameter ζ_y	0.01	-	-
Simulation parameters	steps per revolution	256	-	-

values used in datasets 2 and 3 that are unchanged as compared to dataset 1. Chip width and spindle speed, which range from 0.004 to 0.005 mm and 3000 to 4000 rpm respectively, are the characteristics that vary among the samples in a generated dataset. The produced signals reflect the cutting tool's displacement in the x-direction during its third rotation, sampled at a rate proportionate to the spindle speed.

Each combination of 200 linearly spaced chip width and 200 spindle speed parameter values in the given ranges yields a signal sample, resulting in a total of 40 000 signal samples in the dataset. The only pre-processing done to this data is mean and standard deviation normalisation, which is done individually for each of the process parameters as well as for the time-domain signals. The validation dataset, which includes 40 000 samples, is created using the same method but with the process parameters moved half a step, i.e. chip width ranging from 0.004025 to 0.005025 and spindle speed from 3002.5 to 4002.5. All the datasets are freely accessible on GitLab at `https://gitlab.com/EZotoff/cyclestylegan-based-knowledge-transfer-for-a-machining-digital-twin`.

4.2.2 CycleStyleGAN architecture

Following the style-based modelling approach used for vibration signal synthesis, the proposed CycleStyleGAN generators also operate on the style encodings of the signals. The generators G_{ab} and G_{ba} are thus built as ensembles of three subnetworks: the encoder, the translator and the decoder, and implement the signal translation functions $G_{ab} : x_a \rightarrow x_b$, $G_{ba} : x_b \rightarrow x_a$. The encoder network compresses the input signals into their style representations ($Encoder_{ab} : x_a \rightarrow S_a$ and $Encoder_{ba} : x_b \rightarrow S_b$), which are then passed onto the translator module. This module transforms the received style vector into the target domain style ($Translator_{ab} : S_a \rightarrow S_b$, $Translator_{ba} : S_b \rightarrow S_a$). Finally, the decoder subnetwork synthesises the target domain signal from the translated ($Decoder_{ab} : S_b \rightarrow x_b$, $Decoder_{ba} : S_a \rightarrow x_a$). The schematic

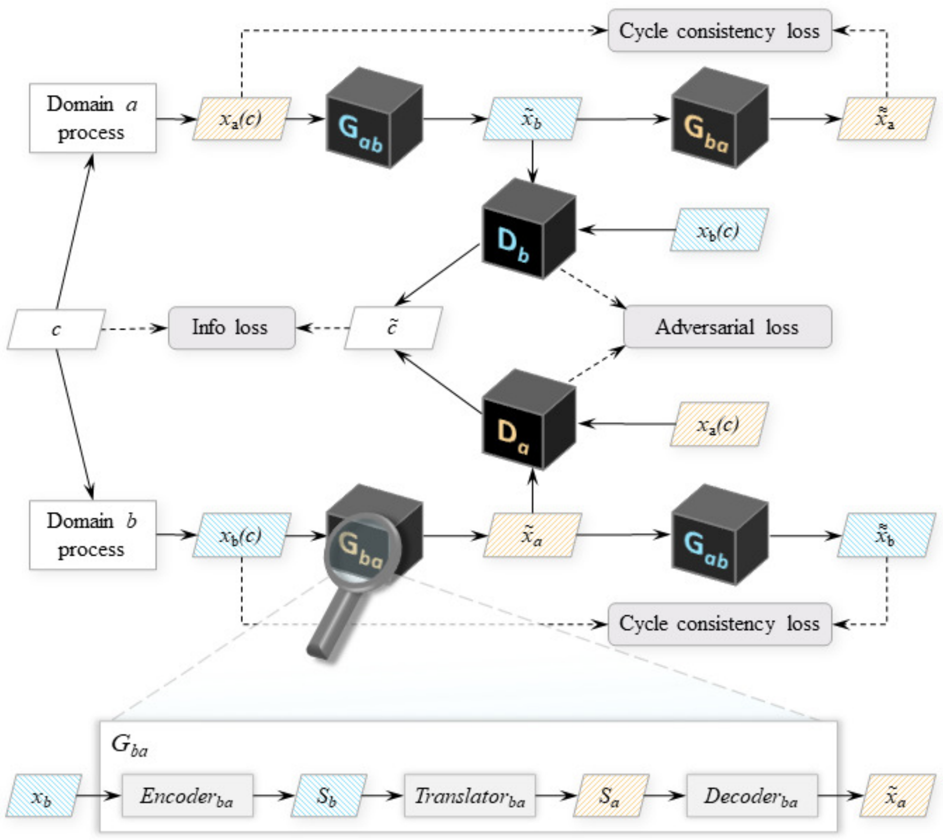


Figure 4.1: **Conditional CycleStyleGAN architecture.** G_{ab} and G_{ba} denote the generator networks that translate signals from domain a to domain b and vice versa. D_a and D_b are the discriminators that process domain a and b signals, both real and the ones translated into their respective domain. The callout on the bottom shows the subnetworks of G, using G_{ba} as the example. Other notation: c, $\tilde{c}$ - real and estimated process parameter values; x_a (x_b) - true domain a (b) signals; $\tilde{x}_a$ ($\tilde{x}_b$) - signals synthesised via translation to domain a (b); $\tilde{\tilde{x}}_a$ ($\tilde{\tilde{x}}_b$) - signals synthesised via reconstruction back to domain a (b); S_a (S_b) - style encoding of a domain a (b) signal. The different colour coding indicates the data structures and the neural networks associated with each domain.

depiction of the CycleStyleGAN architecture is displayed on figure 4.1. The parametrisation of the described networks shall be explained in section 4.2.3.

The current work implements the encoder and the translator networks as deep convolutional networks with residual blocks that each contain two convolutional layers. The decoder subnetwork is similar to the synthesis network described in section 3.2.2 and is a deep transposed convolutional network modified with skip connections (see the synthesis network F block on figure 4.2).

The two discriminators in the CycleStyleGAN model play roles similar to the ones seen in the Conditional StyleGAN model with a key difference in their classification task: the networks now aim to identify whether the signals passed to them belong to their respective domains.

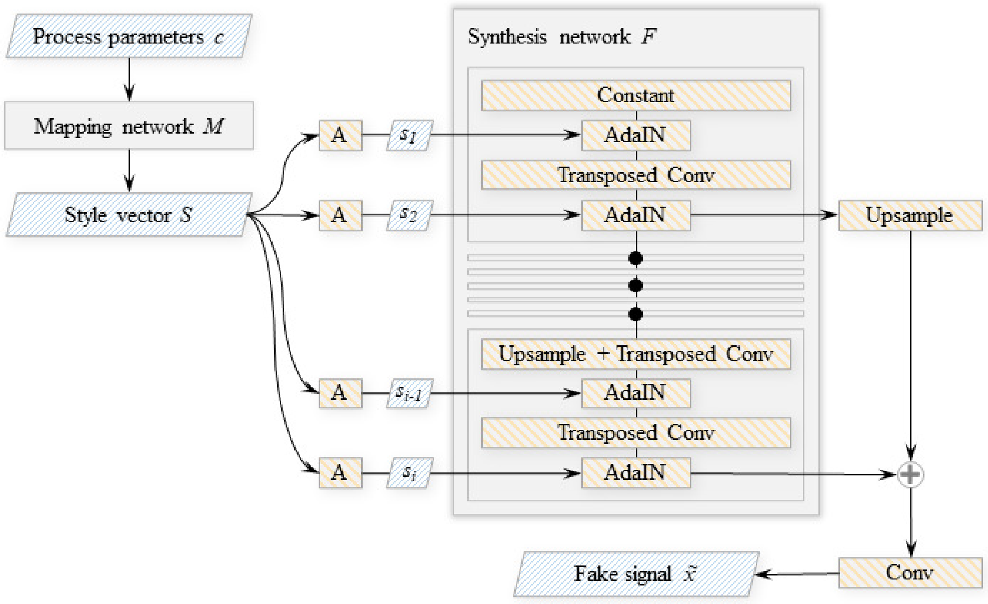


Figure 4.2: **Architecture of the StyleGAN generator network.** "A" denotes learned affine transformations of style components s_i; "AdaIN" - adaptive instance normalisation, outputs of which are modulated by the transformed style components.

The adversarial losses are then formulated as:

$$\begin{aligned}
L_{G_{ab}}^{wgan\text{-}gp} &= -\mathop{\mathbb{E}}_{\tilde{x}_b \sim \mathbb{P}_b}[D_b(\tilde{x}_b)], \\
L_{G_{ba}}^{wgan\text{-}gp} &= -\mathop{\mathbb{E}}_{\tilde{x}_a \sim \mathbb{P}_a}[D_a(\tilde{x}_a)], \\
L_{D_b}^{wgan\text{-}gp} &= \mathop{\mathbb{E}}_{\tilde{x}_b \sim \mathbb{P}_b}[D(\tilde{x}_b)] - \mathop{\mathbb{E}}_{x_a \sim \mathbb{P}_a}[D(x_a)] + \lambda_{gp} L_b^{gp}, \\
L_{D_a}^{wgan\text{-}gp} &= \mathop{\mathbb{E}}_{\tilde{x}_a \sim \mathbb{P}_a}[D(\tilde{x}_a)] - \mathop{\mathbb{E}}_{x_b \sim \mathbb{P}_b}[D(x_b)] + \lambda_{gp} L_a^{gp},
\end{aligned} \tag{4.1}$$

where

$$\begin{aligned}
L_a^{gp} &= \mathop{\mathbb{E}}_{\tilde{x}_a \sim \mathbb{P}_{\tilde{x}_a}}[(||\nabla_{\tilde{x}_a} D(\tilde{x}_a)||_2 - 1)^2] \text{ and} \\
L_b^{gp} &= \mathop{\mathbb{E}}_{\tilde{x}_b \sim \mathbb{P}_{\tilde{x}_b}}[(||\nabla_{\tilde{x}_b} D(\tilde{x}_b)||_2 - 1)^2]
\end{aligned} \tag{4.2}$$

are the gradient penalty terms, λ_{gp} is the gradient penalty scaling hyperparameter, D_a and D_b are the discriminator networks operating on domain a and b signals respectively, x_a (x_b) and $\tilde{x}_a$ ($\tilde{x}_b$) denote the real domain a (b) signals and the signals translated to the domain a (b) respectively and $\mathbb{P}_a$ ($\mathbb{P}_b$) is the domain a (b) signal distributions.

As in the case of the information loss formulated for the Conditional StyleGAN, the generators are incentivised to preserve the label information in the synthesised signals, while the discriminators are penalised for misreading the labels encoded within the training data samples. These targets are implemented as the following loss function for the CycleStyleGAN networks:

$$\begin{aligned}
L_{G_{ab}}^{info} &= \frac{1}{n}\sum_{k=1}^{n} |c_k - \tilde{c}_{k,b,fake}|, \quad & L_{D_b}^{info} &= \frac{1}{n}\sum_{k=1}^{n} |c_k - \tilde{c}_{k,b,real}|, \\
L_{G_{ba}}^{info} &= \frac{1}{n}\sum_{k=1}^{n} |c_k - \tilde{c}_{k,a,fake}|, \quad & L_{D_a}^{info} &= \frac{1}{n}\sum_{k=1}^{n} |c_k - \tilde{c}_{k,a,real}|,
\end{aligned} \tag{4.3}$$

where

$$\begin{aligned} \tilde{c}_{k,a,fake} &= D_a(\tilde{x}_a) \text{ and} \\ \tilde{c}_{k,b,fake} &= D_b(\tilde{x}_b) \end{aligned} \tag{4.4}$$

are the values of parameter k predicted by the respective discriminators based on signals translated to domain a or b.

$$\begin{aligned} \tilde{c}_{k,a,real} &= D_a(x_a) \text{ and} \\ \tilde{c}_{k,b,real} &= D_b(x_b) \end{aligned} \tag{4.5}$$

are the values predicted from a domain a or a domain b real signal, and c_k are the true parameter values.

The total losses the generator and the discriminator networks of the CycleStyleGAN are as follows:

$$\begin{aligned} L_{G_{ab}} &= L_{G_{ab}}^{wgan\text{-}gp} + \lambda_{info} L_{G_{ab}}^{info} + \lambda_{cycle} L^{cycle}, \\ L_{G_{ba}} &= L_{G_{ba}}^{wgan\text{-}gp} + \lambda_{info} L_{G_{ba}}^{info} + \lambda_{cycle} L^{cycle}, \\ L_{D_b} &= L_{D_b}^{wgan\text{-}gp} + \lambda_{info} L_{D_b}^{info}, \\ L_{D_a} &= L_{D_a}^{wgan\text{-}gp} + \lambda_{info} L_{D_a}^{info}, \end{aligned} \tag{4.6}$$

where λ_{info} represents the scaling factor for the label prediction error loss, λ_{cycle} is the cycle consistency loss multiplier. $\lambda_{info} = 10$, $\lambda_{gp} = 10$ and $\lambda_{cycle} = 10$ are used to parameterise the network losses during training.

4.2.3 Hyperparameter optimisation

The neural networks trained during the experiment described in this work have several hyperparameters that configure their internal structure. The description of what these hyperparameters are and how they are optimised to improve the performance of the models is given below. Where possible, reasonable constraints are enforced to maintain the tractability of the hyperparameter search given the available computational resources.

The number of convolutional blocks, the structure of which is described in section 3.2.2, and the number of convolutional filters used in each block are the main hyperparameters that determine the size and complexity of the neural subnetworks. The generative subnetworks, i.e. the synthesis network of the StyleGAN and the decoder of the CycleStyleGAN, receive the number of filters equal to the respective hyperparameter value at the initial convolutional block, that is then downscaled after each block by the filter scaling factor. The minimal number of filters that a block can have is defined via a hyperparameter. The length of the signal is upscaled at the end of each convolutional block, in a way such that the final output signal is of the target length 256. The subnetworks that process the signal in the opposite direction, the discriminators and the CycleStyleGAN encoder, are built in a reverse manner. The number of filters at the last block is determined by the hyperparameter, and this number decreases towards the beginning of the network, while the length of the signal is downscaled after each block from the input's 256. The full list of the hyperparameters is presented in table 4.2.

The size of StyleGAN mapping network, which is a feedforward network with fully connected layers, is configured via its depth (the number of layers) and breadth (number of neurons per layer). These are fixed at 8 and 32 respectively, following the StyleGAN work [55]. The hyperparameters in GAN subnetworks with similar functions are jointly optimised, i.e. the same value is kept between the instances of such hyperparameters in the different networks. Following this approach, the sizes of the style vector output by the mapping network of the StyleGAN model and by the encoder of the CycleStyleGAN are linked. The equivalence of the StyleGAN synthesis and the CycleStyleGAN decoder network architectures is also maintained this way, as well the configurations of both discriminators and the CycleStyleGAN

Table 4.2: Model hyperparameters

Parameter type	**Allowed values**	**Used value**
Optimiser		
Generator optimiser type	SGD, Adam	Adam
Discriminator optimiser type	SGD, Adam	Adam
Generator learning rate	0.01, 0.001, 0.0001	0.001
Discriminator learning rate	0.001, 0.0001, 0.00001	0.0001
Conditional StyleGAN generator		
Mapping network M		
Number of layers	8	8
Neurons per layer	32	32
Style vector size	32, 128, 256, 1024	512
Synthesis network F		
Number of convolutional blocks	7, 5, 2	2
Starting number of convolutional filters	64, 256, 1024	256
Filter number scaling factor	2, 4	2
Minimal filters number	8, 64, 128	8
Conditional StyleGAN Discriminator		
Number of convolutional blocks	7, 5, 2	5
Final number of convolutional filters	64, 256, 1024	64
Filter number scaling factor	2, 4	2
Maximal filters number	128, 512, 1024	512
CycleStyleGAN generator		
Encoder		
Number of convolutional blocks	7, 5, 2	5
Final number of convolutional filters	64, 256, 1024	64
Filter number scaling factor	2, 4	2
Maximal filters number	128, 512, 1024	512
Style vector size	32, 128, 256, 1024	512
Translator		
Number of convolutional blocks	5, 11	5
Number of filters	2, 32, 128	2
Decoder		
Number of convolutional blocks	7, 5, 2	2
Starting number of convolutional filters	64, 256, 1024	256
Filter number scaling factor	2, 4	2
Minimal filters number	8, 64, 128	8
CycleStyleGAN Discriminator		
Number of convolutional blocks	7, 5, 2	5
Final number of convolutional filters	64, 256, 1024	64
Filter number scaling factor	2, 4	2
Maximal filters number	128, 512, 1024	512

encoder. Another hyperparameter search space constraint adopted from previous experiments sets the discriminator optimiser learning rate to the value of one tenth of the learning rate of the generator.

For further optimisation of the hyperparameter search space we start with only the 12 StyleGAN and optimiser hyperparameters, using a single hyperparameter that defines both learning rates as described above. The StyleGAN model is optimised based on dataset 1 using the Hyperband algorithm [67]. This approach implies training many differently parametrised neural network for a few epochs, selection of the subset of best performing ones with their subsequent further training. After multiple iterations of such selection and training, the hyperparameter values of the best-performing network are considered optimal. After StyleGAN hyperparameter optimisation, the remaining non-linked CycleStyleGAN hyperparameters, which are only the two translator subnetwork hyperparameters, are optimised using the same approach. The hyperparameter optimisation at this stage showed the same results both on dataset 2 and dataset 3.

4.2.4 Training schedules

The GANs presented in this study are trained until convergence, or until 68 000 000 sample instances are shown to the networks, each instance representing a single time-series picked from the training dataset. The training data instances are fed to the network during training in batches of 1 000 at a time, cycling through all the non-repeating batches. The rate of improvement of the root-mean-square error (RMSE) metric is measured and averaged over the testing dataset to determine training convergence. Convergence is considered reached if no error reduction is observed over the last 6 800 000 sample instances, i.e. over the last 10% of the maximal total exposure to the training data.

The Conditional StyleGAN model architecture presented in section 3.2.2 is used in two training approaches, for brevity henceforth called retraining and incremental training. Retraining approach implies training of a freshly initialised StyleGAN neural network on a limited set of data from dataset 2 or 3. Incremental training is implemented as a two-stage training schedule. First, a base neural network is trained using all samples available in dataset 1. Second, a neural network initialised with the weights obtained from stage one training (in other words, a copy of the StyleGAN trained on dataset 1) is trained on a given set of samples from datasets 2 or 3. The sets of samples used for training the networks under both approaches are obtained by randomly selecting a fraction of samples from the respective dataset. The percentages of the used samples are 20%, 15%, 10%, 5%, 2%, 0.8%.

The domain adaptation training of the CycleStyleGANs is performed using the full source domain data (i.e., dataset 1) and a subset of the target domain data. The percentages of the used samples are 20%, 15%, 10%, 5%, 2%, 0.8%, 0.2%. The sample sets of datasets 2 and 3 used under this approach are the same as the sets used for incremental training of the StyleGAN described above.

4.3 Results

The analysis presented in this section seeks the validation of the proposed CycleStyleGAN architecture as a knowledge transfer technique under the target domain data scarcity constraint. To this extent, the accuracies of the CycleStyleGAN model instances are compared with the accuracies of the StyleGAN networks, thus presenting the performance of the proposed domain adaptation method against the incremental learning approach. For comparison fairness, the underlying subnetworks of the models are parametrised identically wherever possible, and the models are treated with the same sets of samples during training and are evaluated on the same validation data.

Each trained StyleGAN and CycleStyleGAN model is evaluated by a generative error metric defined as the average of the mean absolute error $\overline{\mathcal{E}}$ between the target signals from the validation data $x(c^{val})$ and the synthesised signals $\tilde{x}$, i.e. the signals created by StyleGAN from parameters:

$$\tilde{x} = G(c^{val}) \tag{4.7}$$

or translated by the $a \rightarrow b$ CycleStyleGAN from the domain a signals:

$$\tilde{x} = G_{ab}(x(c^{val})), \tag{4.8}$$

$$\overline{\mathcal{E}} = \frac{1}{m} \sum_{j=1}^{m} \mathcal{E}(c_j^{val}), \text{ where} \tag{4.9}$$

$$\mathcal{E}(c) = \frac{1}{n} \sum_{i=1}^{n} |x_i(c) - \tilde{x}_i(c)|, \tag{4.10}$$

where m is the number of samples in the validation dataset and n is the signal length. All models are consecutively trained as described in section 4.2.4, starting with the highest fraction of the target dataset, 20%. The experiment for a particular training approach and dataset is interrupted if the obtained error distribution includes any points above the model deficiency threshold. This threshold is inferred from the generative error evaluated on the target domain validation using the StyleGAN model obtained during stage one of the incremental training, i.e. the model trained only on the source domain dataset:

$$\overline{\mathcal{E}}_{thld} = \frac{1}{m} \sum_{j=1}^{m} \mathcal{E}_{thld}(c_j^{val}), \text{ where} \tag{4.11}$$

$$\mathcal{E}_{thld}(c) = \frac{1}{n} \sum_{i=1}^{n} \left| x_i^{target}(c) - \tilde{x}_i^{source}(c) \right|. \tag{4.12}$$

For the sake of limiting the computations required to execute the experiment, and considering that the focus of this work is on the transfer learning with minimal amount of available data, we do not evaluate the models using more than 20% of the target domain data. The error levels of all models differ insignificantly for training runs utilising 15% or more data. The errors are averaged across multiple training runs for each combination of samples.

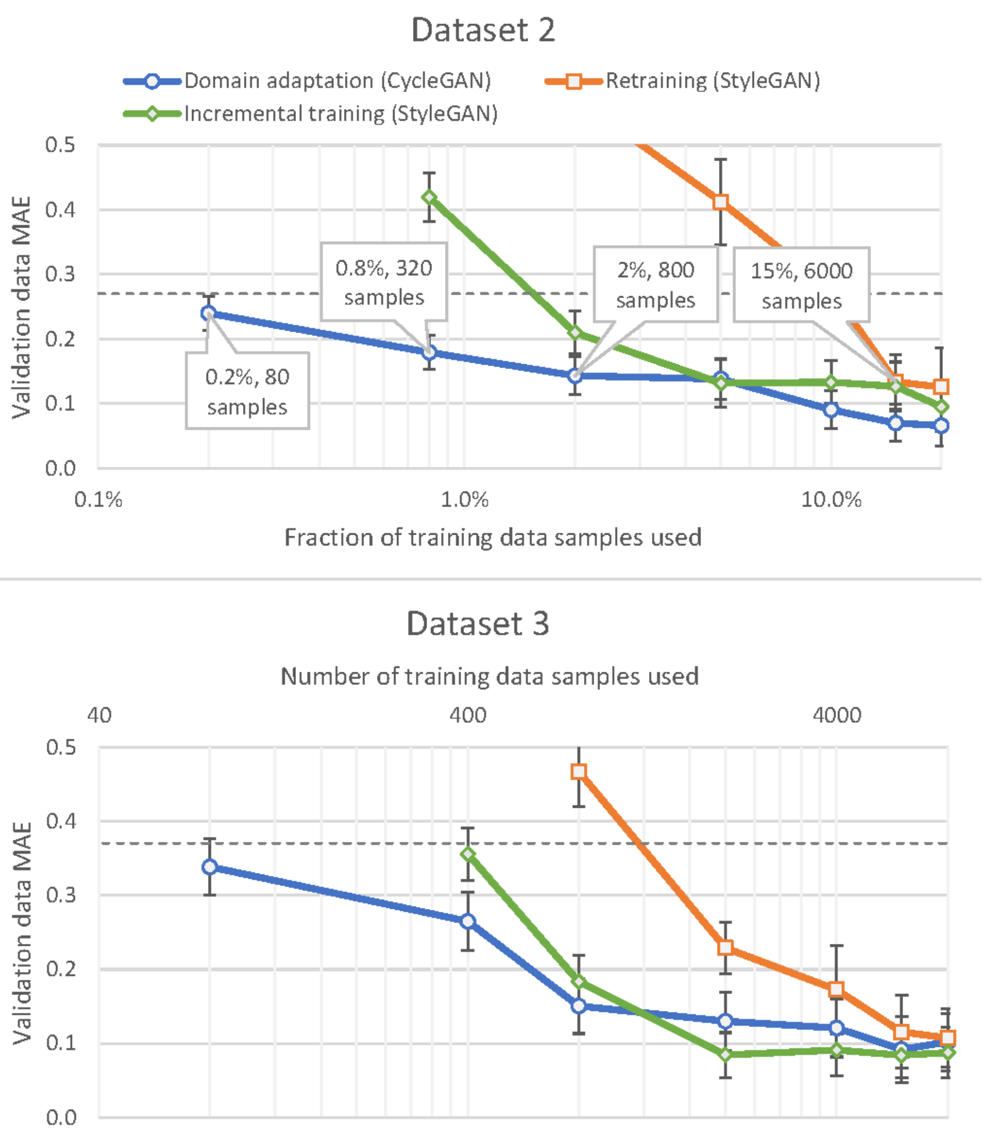


Figure 4.3: **Model error (Y-axis) plotted against the fraction of data used for training (X-axis, log scale) the networks under the three approaches, separated by target domain dataset.** The vertical error bars indicate the standard deviation of the model error across three runs under the same conditions. The dashed horizontal lines on both subcharts represent the errors $\overline{\mathcal{E}}_{thld}$ of the models trained on dataset 1 when evaluated against dataset 2 and 3 validation data respectively. Any error values above the dashed lines thus indicate that the training of a model on the target domain data yields worse results than the application of the source-domain model (i.e. the one trained on dataset 1) used without any exposure to the target domain data.

The training performance distributions of the models at the different levels of target domain data limitations are presented on figure 4.3. Dataset 2 represents a scenario of small difference between the source and the target domains, e.g. as a result of minor variations in material characteristics or environment conditions. Dataset 3 expresses a case of significantly different characteristics underlying the target domain signals, for example arising from a machining tool with a different geometry. A model trained without any source domain knowledge performs well on both datasets when trained using 6 000 (out of the total 40 000) or more target domain training data samples, with smaller training dataset size leading to a sharp drop in the models' performance. The incremental learning approach shows a similar pattern of severe generative accuracy degradation, but below a more strict data limitation constraint: 2 000 samples. The domain adaptation implementation using the CycleStyleGAN architecture proposed in this thesis displays different behaviour to the aforementioned approaches. The quality of the generated signals significantly degrades only when trained on less than 800 target domain data samples, and the degradation below this point is smoothly approaching the $\overline{\mathcal{E}}_{thld}$ error level.

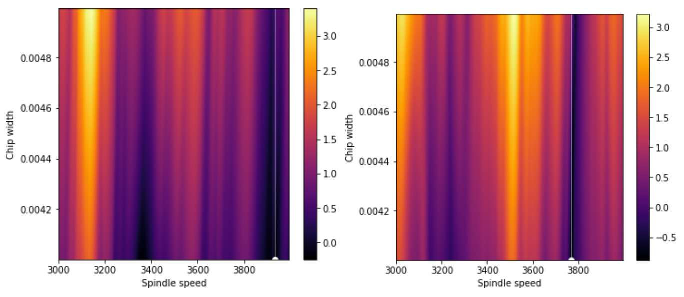
These results imply that the CycleStyleGAN error has an upper bound at $\overline{\mathcal{E}}_{thld}$, the target domain accuracy level of the model trained only on the source domain data. Therefore, the reliability of this model can be estimated from the expected difference between the source and the target domains. The use of the source domain data during CycleStyleGAN training ensures that the model does not suffer from catastrophic forgetting and does not overfit to the small subset of the observed data samples, contrary to what happens to the neural networks trained from scratch or trained incrementally. The CycleStyleGAN domain adaptation is thus potentially usable with any amount of data available at hand at a given moment and can be expected to reach peak performance with the amount of target domain data one order of magnitude lower than a model trained from scratch. For an industrial implementation this means that, on one hand, the value of the knowledge extracted from the source model is not diluted during the knowledge transfer process, and, on the other hand, that the adaptation of the transferred information to a new process can be initiated along with the launch of this process. For a process that requires generative accuracy above the threshold bound, the proposed method enables a reduction of the pre-launch data acquisition effort almost tenfold.

A simple process quality metric may be used for the illustration of a possible application of the proposed model in manufacturing process optimisation. The amplitude of the vibration signal is thus considered here as a crude proxy for the machined product quality due to its computational simplicity and its common association with machining process degradation [4, 60]. Specifically, the peak-to-peak amplitude u of a vibration signal x is measured as the difference between the maximal and the minimal displacement magnitudes of this signal:

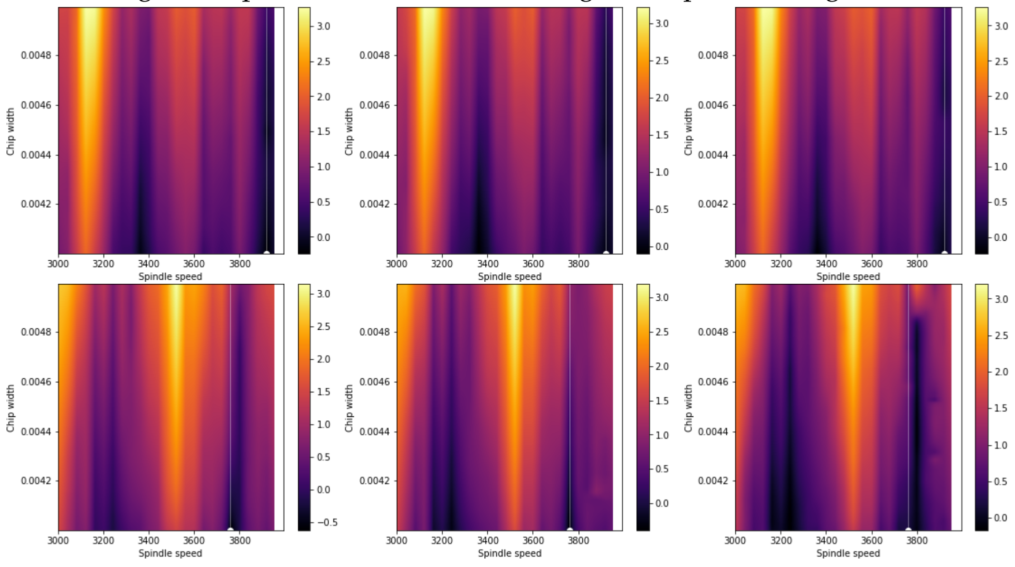
$$u(x) = max(x) - min(x). \tag{4.13}$$

Figure 4.4 depicts the amplitude metric mapped across the chip width and spindle speed value ranges for the two target domains. For both datasets, the process parameters resulting in the minimal vibration amplitude are identified to within 20 spindle speed rpm. Taking into account that the full range of possible spindle speed values in this experiment is 1000 rpm (from 3000 rpm to 4000 rpm), the process parameter optimisation accuracy achieved in this case can be considered to be high. The proposed method is not meant to effectively optimise the machining process parameters, but serves as a short example of a use case scenario that the generative model can be applied within.

It must be noted that the available computational power limitations implied that the number of training repetitions in the described experiment had to be limited to three for each set of training conditions. The server that ran the computations was equipped with Nvidia Titan RTX GPU, which handled most of the neural network training, and two Intel Xeon Silver 4214 CPUs. While this simplification blurs the precision



(a) True signal amplitudes. Left: dataset 2 signal amplitudes, right: dataset 3.



(b) Amplitudes of the predicted signal. Top: dataset 2. Bottom: dataset 3

Figure 4.4: **Amplitudes obtained from the true vibration signals and from the predicted signals** produced by three CycleStyleGAN models for each dataset. White marker indicates the minimal vibration amplitude.

of the estimated model error distributions, the relatively low variation in the models' effectiveness provides sufficient evidence to support the claims presented in the current work. The stability of training of the presented models is empirically supported by the fact that not a single training run resulted in abnormally high error rates.

4.4 Chapter summary

Manufacturing applications of GANs as a primary generative instrument are very limited in the existing research literature, with most works focussing on the data augmentation capacity of GANs, and an even less studied area is the application of digital twin simulations in the context of knowledge extraction for Industry 4.0.

These research gaps are addressed within the work described in this chapter. The style-based representation of the simulated vibration signals has been shown in a previous publication to be useful both for performance improvement and for analysis of the underlying model [132]. Building on the GAN extensions for the transfer learning tasks, namely the CycleGAN architecture, this dissertation introduces the style features into the domain adaptation model via the novel CycleStyleGAN architecture.

The analysis of the performance of the CycleStyleGAN-based domain adaptation model proposed here displays the tenfold data efficiency gain attainable due to the extraction of the knowledge from a data-abundant domain as well as a smoother model accuracy degradation associated with the data availability constraint compared to the two naïve approaches. This implies more realistic requirements for the application of a machining digital twin at an industrial scale. The following chapter reviews this problem in the context of Industry 4.0 and proposes a use case for the knowledge transfer tool described in this chapter.

Chapter 5

Conclusion

5.1 Summary of research and contributions

The first application of a style-based GAN for machining process simulation is presented in this thesis. The work uses both the style and conditional components of the GAN architecture for attainment of deeper insights into the generative function of the neural network. A novel sensitivity analysis approach for conditional GANs is presented and applied to the proposed model, establishing a link between the true accuracy and the label sensitivity of the generator network. The style component of the GAN architecture is utilised to further advance this analysis to the level of intermediate neural network layers, resulting in the identification of the activation nodes that produce abnormal generative behaviour.

A purely support role of the generator is considered in most relevant publications in the manufacturing domain. Being one of the first few works that focus on the generator of GAN, the current thesis aims to promote the use of GANs in manufacturing past simple data augmentation for imbalanced datasets towards more sophisticated simulation applicable to a wider range of use cases. The development of the CycleStyleGAN model for knowledge transfer further contributes to this aim by proposing a solution that, on one hand, reuses the value of the available simulation modelling resources or know-how and, on the other hand, makes the model applicable to the deployment cases constrained by severe data limitation.

The contributions of the work presented here can thus be summarised in the following list.

1. A novel StyleGAN-based data-driven machining simulation model is developed to be used as a digital twin component.
2. Knowledge extraction capabilities of the proposed StyleGAN model are demonstrated.
3. A data-agnostic sensitivity-based GAN evaluation approach is proposed and its relation to the model's generative quality is presented.
4. A novel CycleStyleGAN architecture is proposed and the evidence of its effectiveness under data availability constraints is presented.

In a real-world scenario a physics-based model would be the same for a fleet of machines, but each machine would be operated in different conditions and have slightly varying characteristics. The calibration of such physics-based models is time-consuming and difficult to perform on a large scale. Therefore, while a physics-based model might act as a digital twin simulation component, it would inevitably have made simplifications and idealisations about the process, likely omitting individual variations of environmental and dynamic factors that influence the manufacturing process due to their modelling complexity or computational cost. These complex phenomena are nevertheless reflected in the real process data and can thus be captured via data-driven training of a GAN model. The physics-based model used in the presented work acts to prepare a surrogate for the real-world data that a digital twin is likely to have access to. The current work thus shows not only a data-driven simulation modelling approach, but

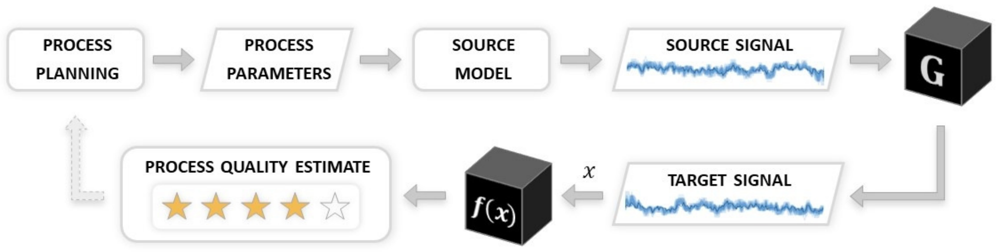


Figure 5.1: Proposed generator model as a part of a process optimisation flow.

also a mechanism for the incorporation of the knowledge contained within a physics-based model into the data-driven modelling process. Thus, the proposed method not only generates value from data, but also effectively extracts the highly valuable knowledge from the existing physics- or expert-based simulation tools.

Control over the input machining process parameters guides the process signal synthesis in the proposed CycleStyleGAN model. Therefore, the model may serve as a vibration simulation tool that translates the process parameter inputs into vibration signal outputs when combined with a source domain simulation model that produces source signals from process parameters. Such a simulation may be used for CNC machining process optimisation and planning. Researchers have shown how signal data could be utilised for product quality prediction [44, 61, 85], enabling the prediction of manufacturing defects prior to manufacturing. Furthermore, machining process stability estimates may be improved by substituting generative models for parts of the physical data, which is another future study path suggested by machining stability specialists [39]. The proposed CycleStyleGAN model is thus usable as a process optimisation tool: by probing the model to acquire parameter-signal pairs and evaluating the resultant process quality based on the received signals, an optimisation process loop would seek the optimum within the parameter space. Schematic representation of this process flow is depicted on figure 5.1. This thesis shows how an established physics-based or data-driven model can be sourced, thus extracting the value of the information contained within the said model for further reuse. The cost-efficiency of the proposed model is believed to be an important driver towards the widespread use of digital twin solutions along the transition to Industry 4.0.

5.2 Model limitations and further research

The current work considers a conditional GAN with only continuous inputs, but an adaptation of the described sensitivity analysis method to the conditional GAN models with categorical inputs would make the proposed technique universally applicable in GAN performance analysis. Altering the neural network architecture could also make further insights available by further shifting the complexity of the multi-layer convolutional interactions to other operations, such as via skip connections [48] or residual connections [43].

Future research directions include comparative evaluation of different neural network architectures and validation of the proposed model on real manufacturing data, as well as broadening of the scope of the digital twin simulation with inclusion of multiple data sources and simultaneously modelled processes.

Hybrid machining simulation models could be further developed by fusing the physics-based data with the sensor readings collected from a live process. The machining datasets composed may be useful for future work in this direction and are available via `https://gitlab.com/EZotoff/conditional-stylegan-digital-twin` for the dataset used in chapter 3 or at `https://gitlab.com/EZotoff/cyclestylegan-based-knowledge-transfer-for-a-machining-digital-twin` for the datasets composed for the experiments presented in chapter 4.

A drawback of the CycleStyleGAN model architecture, as compared to the StyleGAN, is its higher computational resource requirements. The mirrored GAN structure uses approximately double the memory and double the computations during training. While these differences are negligible for a trained model due to the efficiency of the neural networks at inference time, the training process computations are twice as costly. Although the costs of computational hardware are incomparable to the manufacturing process expenses in most cases, certain high-volume low-value production industries might find the computational overhead exceeding the expected value of such simulation model adaptation. Such businesses, having relatively lower data acquisition costs, can be expected to be able to effectively employ machine learning models without the need for transfer learning.

Possible extensions of the proposed domain adaptation approach may consider inter-task transfer learning. For example, the prediction of the machining cutting forces from the vibration signals may useful for downstream process analysis. Another research gap the exploration of which might lead to improved generative performance of the underlying model is the specialised subnetwork architecture. The current literature presents multiple options for structuring of these neural networks, but a comparative analysis of the performance of these architecture choices is yet to reach the science community. Linked to this is the application of sophisticated neural network architecture search approaches. An extension of an advanced method like the ES-HyperNEAT [91] might aid not only the hyperparameter optimisation of a pre-defined network structure, but also in discovery of a novel composition of the neural network.

The practical goal of these research endeavours would be the application of the data-driven modelling and simulation tools for a highly autonomous manufacturing analytics and optimisation suite, such as the one concisely depicted on fig. 5.2. And while the computational and decision-making abilities necessary to run such a technological ensemble on industrial scale seems to currently lie beyond the capabilities of the existing AIs, the work presented in this thesis intends to advance the scientific thought towards their wide applicability among the manufacturing enterprises.

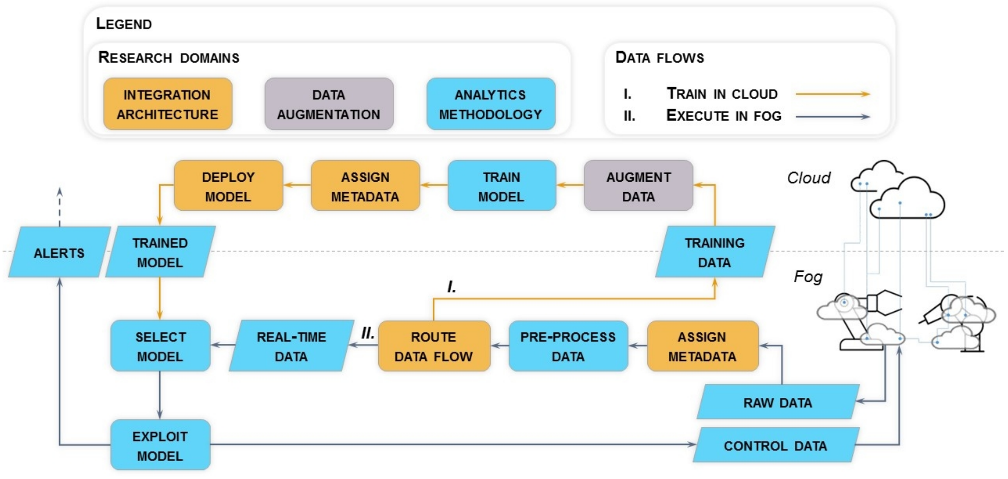


Figure 5.2: Target manufacturing analytics process flow

Bibliography

[1] S.M. Afazov, S.M. Ratchev, and J. Segal. Modelling and simulation of micro-milling cutting forces. *Journal of Materials Processing Technology*, 210(15):2154–2162, 2010.

[2] D. Agrawal, S. Das, and A. El Abbadi. Big data and cloud computing. In *Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT '11*, page 530, New York, USA, 2011.

[3] M.B. Alawieh, Y. Lin, Z. Zhang, M. Li, Q. Huang, and D.Z. Pan. GAN-SRAF: Sub-resolution assist feature generation using conditional generative adversarial networks. *Proceedings - Design Automation Conference*, (i):1–6, 2019.

[4] Y. Altintas and M. Weck. Chatter stability of metal cutting and grinding. *CIRP Annals - Manufacturing Technology*, 53(2):619–642, 2004.

[5] Y. Altintas, P. Kersting, D. Biermann, E. Budak, B. Denkena, and I. Lazoglu. Virtual process systems for part machining operations. *CIRP Annals - Manufacturing Technology*, 63(2):585–605, 2014.

[6] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN, 2017. URL `http://arxiv.org/abs/1701.07875`.

[7] S. Bai, J.Z. Kolter, and V. Koltun. An empirical evaluation of

generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[8] M. Bajaj, D. Zwemer, and B. Cole. Architecture to geometry - Integrating system models with mechanical design. *AIAA Space and Astronautics Forum and Exposition, SPACE 2016*, pages 1–19, 2016.

[9] S.H. Bang, R. Ak, A. Narayanan, Y.T. Lee, and H. Cho. A survey on knowledge transfer for manufacturing data analytics. *Computers in Industry*, 104:116–130, 2019.

[10] Y. Bengio. *Learning deep architectures for AI*, volume 2. 2009.

[11] W. Booyse, D.N. Wilke, and S. Heyns. Deep digital twins for detection, diagnostics and prognostics. *Mechanical Systems and Signal Processing*, 140:106612, 2020.

[12] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:95–104, 2017.

[13] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250. may 2018.

[14] A. Brock, J. Donahue, and K. Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. sep 2018.

[15] M.L. Campomanes and Y. Altintas. An improved time domain simulation for dynamic milling at small radial immersions. *Jour-*

nal of Manufacturing Science and Engineering, Transactions of the ASME, 125(3):416–422, 2003.

[16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In D. Cremers, I. Reid, H. Saito, and M.H. Yang, editors, *Proceedings of the 30th International Conference on Neural Information Processing Systems*, volume 9006 of *Lecture Notes in Computer Science*, pages 2180–2188, Cham, jun 2016.

[17] Y. Chen, Y. Wang, D. Kirschen, and B. Zhang. Model-Free Renewable Scenario Generation Using Generative Adversarial Networks. *IEEE Transactions on Power Systems*, 33(3):3265–3275, may 2018.

[18] H. Cherukuri, E. Perez-Bernabeu, M. Selles, and T. Schmitz. Machining chatter prediction using a data learning model. *Journal of Manufacturing and Materials Processing*, 3(2), 2019.

[19] A. Church and A.M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *The Journal of Symbolic Logic*, 2(1):42, 1937.

[20] D.C. Ciresan, U. Meier, L.M. Gambardella, and J. Schmidhuber. Convolutional Neural Network Committees for Handwritten Character Classification. In *2011 International Conference on Document Analysis and Recognition*, pages 1135–1139. sep 2011.

[21] E. Conti, V. Madhavan, F.P. Such, J. Lehman, K.O. Stanley, and J. Clune. Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. pages 1–17, 2017.

[22] C. Cooper, J. Zhang, R.X. Gao, P. Wang, and I. Ragai. Anomaly detection in milling tools using acoustic signals and generative

adversarial networks. *Procedia Manufacturing*, 48(2019):372–378, 2020.

[23] B. Ding, H. Qian, and J. Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, volume 2, pages 1836–1841. jun 2018.

[24] C. Donahue, J. McAuley, and M. Puckette. Adversarial Audio Synthesis. In *International Conference on Learning Representations*, pages 1–16, feb 2019.

[25] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning, 2018. URL `https://arxiv.org/abs/1603.07285`.

[26] W. Elmaraghy, H. Elmaraghy, T. Tomiyama, and L. Monostori. Complexity in engineering design and manufacturing. *CIRP Annals - Manufacturing Technology*, 61(2):793–814, 2012.

[27] T. Elsken, J.H. Metzen, and F. Hutter. Neural Architecture Search. *Journal of Machine Learning Research*, 20:1—-21, 2019.

[28] J. Engel, K.K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts. Gansynth: Adversarial neural audio synthesis. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–17, 2019.

[29] EPSRC Future Metrology Hub. UK Metrology Research Roadmap. Technical report, 2020.

[30] C. Esteban, S.L. Hyland, and G. Rätsch. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs, jun 2017. URL `http://arxiv.org/abs/1706.02633`.

[31] A.B. Forbes. Uncertainty associated with form assessment in coordinate metrology. *International Journal of Metrology and Quality Engineering*, 4(1):17–22, 2013.

[32] J. Friedrich, J. Torzewski, and A. Verl. Online Learning of Stability Lobe Diagrams in Milling. *Procedia CIRP*, 67:278–283, 2018.

[33] K. 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.

[34] Y. Ganin, H. Larochelle, and M. Marchand. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35, 2016.

[35] Y. Gao, R. Singh, and B. Raj. Voice Impersonation Using Generative Adversarial Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018-April: 2506–2510, 2018.

[36] C. Giraud-carrier. A Note on the Utility of Incremental Learning. *AI Communications*, (February), 2013.

[37] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 15:315–323, 2011.

[38] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks, 2014. URL `http://arxiv.org/abs/1406.2661`.

[39] N.P. Greis, M.L. Nogueira, S. Bhattacharya, and T. Schmitz. Physics-Guided Machine Learning for Self-Aware Machining. In *2020 AAAI Spring Symposium on AI and Manufacturing*, 2020.

[40] M. Grieves and J. Vickers. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, pages 85–113. Springer International Publishing, 2017.

[41] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5769–5779, Red Hook, NY, USA, 2017.

[42] T. Han, C. Liu, W. Yang, and D. Jiang. A novel adversarial learning framework in deep convolutional neural network for intelligent diagnosis of mechanical faults. *Knowledge-Based Systems*, 165:474–487, 2019.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *Multimedia Tools and Applications*, pages 1–17, dec 2015.

[44] Q.P. He and J. Wang. Statistical process monitoring as a big data analytics tool for smart manufacturing. *Journal of Process Control*, 67:35–43, 2018.

[45] K. Henning, W. Wolfgang, and H. Johannes. Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Technical Report April, 2013.

[46] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[47] J. Hoffman, E. Tzeng, T. Park, J.Y. Zhu, P. Isola, K. Saenko, A.A. Efros, and T. Darrell. CyCADA: Cycle-Consistent Adversarial Domain adaptation. *35th International Conference on Machine Learning, ICML 2018*, 5:3162–3174, 2018.

[48] G. Huang, Z. Liu, L. Van Der Maaten, and K.Q. Weinberger. Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:2261–2269, 2017.

[49] X. Huang and S. Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, mar 2017.

[50] A. Issa, D. Lucke, and T. Bauernhansl. Mobilizing SMEs Towards Industrie 4.0-enabled Smart Products. *Procedia CIRP*, 63:670–674, 2017.

[51] J. Jiao, M. Zhao, J. Lin, and K. Liang. A comprehensive review on convolutional neural network in machine fault diagnosis. *Neurocomputing*, 417:36–63, 2020.

[52] M.I. Jordan and T.M. Mitchell. Machine learning: Trends, perspectives, and prospects. 349(6245), 2015.

[53] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S.A. Kohl, A.J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A.W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[54] L. Juvela, B. Bollepalli, X. Wang, H. Kameoka, M. Airaksinen, J. Yamagishi, and P. Alku. Speech Waveform Synthesis from MFCC Sequences with Generative Adversarial Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018-April:5679–5683, 2018.

[55] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[56] T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, dec 2018.

[57] D.H. Kim, T.J. Kim, X. Wang, M. Kim, Y.J. Quan, J.W. Oh, S.H. Min, H. Kim, B. Bhandari, I. Yang, and S.H. Ahn. Smart Machining Process Using Machine Learning: A Review and Perspective on Machining Industry. *International Journal of Precision Engineering and Manufacturing - Green Technology*, 5(4):555–568, 2018.

[58] D.P. Kingma and J.L. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, pages 1–15, 2015.

[59] A. Kusiak. Convolutional and generative adversarial neural networks in manufacturing. *International Journal of Production Research*, 0(0):1–11, 2019.

[60] C.H. Lauro, L.C. Brandão, D. Baldo, R.A. Reis, and J.P. Davim. Monitoring and processing signal applied in machining processes - A review. *Measurement: Journal of the International Measurement Confederation*, 58:73–86, 2014.

[61] M. Leco and V. Kadirkamanathan. A perturbation signal based data-driven gaussian process regression model for in-process part quality prediction in robotic countersinking operations. *Robotics and Computer-Integrated Manufacturing*, 71:102105, 10 2021.

[62] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[63] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

[64] J. Lee, E. Lapira, B. Bagheri, and H. an Kao. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, 1(1):38–41, 2013.

[65] P. Lennie. The Cost of Cortical Computation. *Current Biology*, 13 (6):493–497, mar 2003.

[66] C. Li, S. Zhang, Y. Qin, and E. Estupinan. A systematic review of deep transfer learning for machinery fault diagnosis. *Neurocomputing*, 407:121–135, sep 2020.

[67] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18 (1):6765–6816, January 2017.

[68] Y. Li, C. Liu, J.X. Gao, and W. Shen. An integrated feature-based dynamic control system for on-line machining, inspection and monitoring. *Integrated Computer-Aided Engineering*, 22(2):187–200, 2015.

[69] M.y. Liu and O. Tuzel. Coupled Generative Adversarial Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 469–477, 2016.

[70] A.L. Maas, A.Y. Hannun, and A.Y. Ng. Rectifier nonlinearities improve neural network acoustic models. *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 28, 2013.

[71] J. McCarthy, M.L. Minsky, N. Rochester, and C.E. Shannon. A proposal for the Dartmouth summer research project on artificial intelligence, August 31, 1955. *AI Magazine*, 27(4):12–14, 2006.

[72] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943.

[73] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. In *In NIPS 2014 Workshop on Deep Learning*, pages 1–7, nov 2014.

[74] O. Mogren. C-RNN-GAN: continuous recurrent neural networks with adversarial training. *CoRR*, abs/1611.09904, 2016.

[75] L. Monostori. AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing. *Engineering Applications of Artificial Intelligence*, 16(4):277–291, 2003.

[76] G. Morse and K.O. Stanley. Simple Evolutionary Optimization Can Rival Stochastic Gradient Descent in Neural Networks. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*, pages 477–484, 2016.

[77] E. Mueller, X.L. Chen, and R. Riedel. Challenges and Requirements for the Application of Industry 4.0: A Special Insight with the Usage of Cyber-Physical System. *Chinese Journal of Mechanical Engineering (English Edition)*, 30(5):1050–1057, 2017.

[78] E. Negri, L. Fumagalli, and M. Macchi. A Review of the Roles of Digital Twin in CPS-based Production Systems. *Procedia Manufacturing*, 11(June):939–948, 2017.

[79] W. Nie, N. Narodytska, and A. Patel. RelGAN: Relational Generative Adversarial Networks for Text Generation. In *ICLR*, pages 1–20, 2019.

[80] O. Niggemann, G. Biswas, J.S. Kinnebrew, H. Khorasgani, S. Volgmann, and A. Bunte. Data-driven monitoring of cyber-physical systems leveraging on big data and the internet-of-things for diagnosis and control. *CEUR Workshop Proceedings*, 1507:185–192, 2015.

[81] I. Oleaga, C. Pardo, J.J. Zulaika, and A. Bustillo. A machine-learning based solution for chatter prediction in heavy-duty

milling machines. *Measurement: Journal of the International Measurement Confederation*, 128(May):34–44, 2018.

[82] T. Özel and T. Altan. Process simulation using finite element method — prediction of cutting forces, tool stresses and temperatures in high-speed flat end milling. *International Journal of Machine Tools and Manufacture*, 40(5):713–738, apr 2000.

[83] S.J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[84] M. Papananias, T.E. McLeay, M. Mahfouf, and V. Kadirkamanathan. A Bayesian framework to estimate part quality and associated uncertainties in multistage manufacturing. *Computers in Industry*, 105(February):35–47, 2019.

[85] M. Papananias, T.E. McLeay, M. Mahfouf, and V. Kadirkamanathan. An intelligent metrology informatics system based on neural networks for multistage manufacturing processes. *Procedia CIRP*, 82(June):444–449, 2019.

[86] D.T. Pham and A.A. Afify. Machine-learning techniques and their applications in manufacturing. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 219(5):395–412, 2005.

[87] F.J. Pontes, J.R. Ferreira, M.B. Silva, A.P. Paiva, and P.P. Balestrassi. Artificial neural networks for machining processes surface roughness modeling. *International Journal of Advanced Manufacturing Technology*, 49(9-12):879–902, 2010.

[88] PwC. Industry 4.0: Building the digital enterprise, 2016. URL `https://www.pwc.com/gx/en/industries/industries-4.0/landing-page/industry-4.0-building-your-digital-enterprise-april-2016.pdf`.

[89] X. Qi, G. Chen, Y. Li, X. Cheng, and C. Li. Applying Neural-Network-Based Machine Learning to Additive Manufacturing: Current Applications, Challenges, and Future Perspectives. *Engineering*, 5(4):721–729, 2019.

[90] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learning Representations, ICLR 2016*, pages 1–16, 2015.

[91] S. Risi and K.O. Stanley. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial Life*, 18(4):331–363, 2012.

[92] F. Rosenblatt. The Perceptron - A Perceiving and Recognizing Automaton, 1957.

[93] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Cornell Aeronautical Laboratory. Report no. VG-1196-G-8. Spartan Books, 1962.

[94] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986.

[95] T.L. Schmitz and K.S. Smith. *Machining Dynamics*. Springer International Publishing, Cham, 2019.

[96] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang. DRAFT Modeling, Simulation, information Technology & Processing Roadmap - Technology Area 11. *National Aeronautics and Space Administration*, page 27, 2010.

[97] M. Shahbazi, Z. Huang, D.P. Paudel, A. Chhatkuli, and L. Van Gool. Efficient Conditional GAN Transfer with Knowledge Propagation across Classes. In *2021 IEEE/CVF Conference on Com-*

puter Vision and Pattern Recognition (CVPR), pages 12162–12171. jun 2021.

[98] N. Shetty, S.M. Shahabaz, S.S. Sharma, and S. Divakara Shetty. A review on finite element method for machining of composite materials. *Composite Structures*, 176:790–802, 2017.

[99] S.A. Shevchik, C. Kenel, C. Leinenbach, and K. Wasmer. Acoustic emission for in situ quality monitoring in additive manufacturing using spectral convolutional neural networks. *Additive Manufacturing*, 21:598–604, 2018.

[100] X. Shi, Z. Chen, H. Wang, D.Y. Yeung, W.K. Wong, and W.C. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, 2015-Janua:802–810, 2015.

[101] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 2242–2251, 2017.

[102] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[103] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, oct 2017.

[104] S. Smith and J. Tlusty. Overview of modeling and simulation of the milling process. *Journal of engineering for industry*, 113(2): 169–175, 1991.

[105] A. Spurr, E. Aksan, and O. Hilliges. Guiding InfoGAN with Semi-supervision. In M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 119–134. Springer International Publishing, Cham, jul 2017.

[106] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, jan 2014.

[107] F.P. Such, V. Madhavan, E. Conti, J. Lehman, K.O. Stanley, and J. Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *CoRR*, abs/1712.06567, 2017.

[108] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2058–2065. 2016.

[109] H. Tan, X. Liu, M. Liu, B. Yin, and X. Li. KT-GAN: Knowledge-Transfer Generative Adversarial Network for Text-to-Image Synthesis. *IEEE Transactions on Image Processing*, 30(8):1275–1290, 2021.

[110] R.K. Tan, N.L. Zhang, and W. Ye. A deep learning–based method for the design of microstructural materials. *Structural and Multidisciplinary Optimization*, pages 1–22, 2019.

[111] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui. Digital twin-driven product design, manufacturing and service with big

data. *International Journal of Advanced Manufacturing Technology*, 94(9-12):3563–3576, 2018.

[112] A. Theorin, K. Bengtsson, J. Provost, M. Lieder, C. Johnsson, T. Lundholm, and B. Lennartson. An event-driven manufacturing information system architecture for Industry 4.0. *International Journal of Production Research*, 55(5):1297–1311, 2017.

[113] T. Thepsonthi and T. Özel. 3-D finite element process simulation of micro-end milling Ti-6Al-4V titanium alloy: Experimental validations on chip flow and tool wear. *Journal of Materials Processing Technology*, 221:128–145, 2015.

[114] K. Tidriri, N. Chatti, S. Verron, and T. Tiplica. Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. *Annual Reviews in Control*, 42:63–81, 2016.

[115] H.H. Tseng, Y. Luo, S. Cui, J.T. Chien, R.K. Ten Haken, and I.E. Naqa. Deep reinforcement learning for automated radiation adaptation in lung cancer. *Medical Physics*, 44(12):6690–6705, dec 2017.

[116] A.M. Turing. Computing Machinery and Intelligence. *Mind*, 59 (236):433–460, oct 1950.

[117] D.V. Vargas, J. Murata, H. Takano, and A.C.B. Delbem. General Subpopulation Framework and Taming the Conflict Inside Populations. *Evolutionary Computation*, 23(1):1–36, mar 2015.

[118] V.S. Vishnu, K.G. Varghese, and B. Gurumoorthy. A Data-driven Digital Twin of CNC Machining Processes for Predicting Surface Roughness. *Procedia CIRP*, 104:1065–1070, 2021.

[119] J. Wang, Y. Ma, L. Zhang, R.X. Gao, and D. Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, 48:144–156, 2018.

[120] J. Wang, Z. Yang, J. Zhang, Q. Zhang, and W.T.K. Chien. AdaBalGAN: An Improved Generative Adversarial Network with Imbalanced Learning for Wafer Defective Pattern Recognition. *IEEE Transactions on Semiconductor Manufacturing*, 32(3):310–319, 2019.

[121] Y. Wang, K. Li, S. Gan, C. Cameron, and M. Zheng. Data augmentation for intelligent manufacturing with generative adversarial framework. *1st International Conference on Industrial Artificial Intelligence, IAI 2019*, pages 1–6, 2019.

[122] Z. Wang, J. Wang, and Y. Wang. An intelligent diagnosis scheme based on generative adversarial learning deep neural networks and its application to planetary gearbox fault pattern recognition. *Neurocomputing*, 310:213–222, 2018.

[123] K. Weiss, T.M. Khoshgoftaar, and D. Wang. *A survey of transfer learning*. Springer International Publishing, 2016.

[124] L. Wen, X. Li, L. Gao, and Y. Zhang. A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method. *IEEE Transactions on Industrial Electronics*, 65(7):5990–5998, 2018.

[125] H.P. Wiendahl and P. Scholtissek. Management and Control of Complexity in Manufacturing. *CIRP Annals - Manufacturing Technology*, 43(2):533–540, 1994.

[126] R.G. Wilhelm, R. Hocken, and H. Schwenke. Task specific uncertainty in coordinate measurement. *CIRP Annals - Manufacturing Technology*, 50(2):553–563, 2001.

[127] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473, 2016.

[128] H. Zhang, T. Xu, and H. Li. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks.

In *2017 IEEE International Conference on Computer Vision (ICCV)*, volume 2017-Octob, pages 5908–5916. oct 2017.

[129] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R.X. Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019.

[130] H. Zheng, R. Wang, Y. Yang, J. Yin, Y. Li, Y. Li, and M. Xu. Cross-Domain Fault Diagnosis Using Knowledge Transfer Strategy: A Review. *IEEE Access*, 7:129260–129290, 2019.

[131] J.Y. Zhu, T. Park, P. Isola, and A.A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, volume October, pages 2242–2251. oct 2017.

[132] E. Zotov, A. Tiwari, and V. Kadirkamanathan. Conditional StyleGAN modelling and analysis for a machining digital twin. *Integrated Computer-Aided Engineering*, pages 1–17, jul 2021.