In-situ Tailoring of Functional Properties for the LPBF-Process of Pure Copper

In-Situ Anpassung funktioneller Werkstoffeigenschaften für den LPBF-Prozess mit Reinkupfer

Master-Thesis

Laura Luran Sun

Wirtschaftsingenieurwesen Maschinenbau

TECHNISCHE UNIVERSITÄT DARMSTADT

pmd

produktentwicklung maschinenelemente

Laura Luran Sun
Studiengang: Wirtschaftsingenieurwesen Maschinenbau

Masterarbeit
Thema: In-situ Tailoring of Functional Properties for the LPBF-Process of Pure Copper

Eingereicht: 15.04.2024

Betreuer: Moritz Schäfle, M.Sc.

Fachgebiet Produktentwicklung und Maschinenelemente
Fachbereich Maschinenbau
Technische Universität Darmstadt
Otto-Berndt-Str. 2
64287 Darmstadt

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Laura Luran Sun, die vorliegende Master-Thesis / Bachelor-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

English translation for information purposes only:

Thesis Statement pursuant to § 22 paragraph 7 and § 23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Laura Luran Sun, have written the submitted thesis independently pursuant to § 22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware, that in case of an attempt at deception based on plagiarism (§ 38 paragraph 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

In the submitted thesis the written copies and the electronic version for archiving are pursuant to § 23 paragraph 7 of APB identical in content.

For a thesis of the Department of Architecture, the submitted electronic version corresponds to the presented model and the submitted architectural plans.

Datum / Date: Unterschrift / Signature:

_____ _____

Masterarbeit

für

Frau Laura Luran Sun

Matr.-Nr. 2474760

TECHNISCHE UNIVERSITÄT DARMSTADT

In-situ Anpassung funktioneller Werkstoffeigenschaften für den LPBF-Prozess mit Reinkupfer

In-situ Tailoring of Functional Properties for the LPBF-Process of Pure Copper

Mittels der im LPBF-Prozess veränderbaren Mikrostruktur metallischer Werkstoffe ist eine voxelbasierte Anpassung von mechanischen, sowie funktionellen Eigenschaften im µm-Bereich möglich. Besonders für Werkstoffe wie Kupfer, welche in Anwendungen eingesetzt werden, welche hohe Ansprüche an die mechanischen und elektrischen Eigenschaften stellen, ist diese Möglichkeit hoch relevant. Perspektivisch lassen sich bisher unerreichte Kombinationen von Eigenschaften erzielen, welche sich in einem homogenen Bauteil diametral gegenüberstehen. Die alternative Herstellung von Bauteilen mit gleicher oder ähnlicher Funktionalität erfordert unter Umständen die Kombination mehrerer Werkstoffe, das Fügen mehrerer Bauteile oder ist in einer vergleichbaren Qualität nicht möglich.

Basierend auf Vorarbeiten zur Einstellung eines Prozesses zur Herstellung von Bauteilen hoher Dichte werden bestehende Ansätze erweitert, um Bauteileigenschaften bei gleichbleibend hoher Dichte zu variieren. Im Fokus stehen dabei mechanische und elektrische Eigenschaften. Die dargelegte Forschungsaufgabe wird durch den Einsatz analytischer Methoden und künstlicher neuronaler Netze unterstützt.

- Darlegung des Stands der Forschung für LPBF von Kupfer und der Beeinflussung der Eigenschaften metallischer Werkstoffe im LPBF-Prozess, sowie möglicher Anwendungsgebiete
- Untersuchung der Möglichkeit zur Veränderung mechanischer und elektrischer Eigenschaften zwischen Bauteilen und Bauteilsektionen durch Anpassung der Parameter des LPBF-Prozesses
- Analyse der erzielten mechanischen und elektrischen Bauteileigenschaften und Darlegung der erkannten Prozess-Eigenschafts-Beziehungen
- Weiterentwicklung bestehender Ansätze zur schlanken Prozessentwicklung durch analytische und KI-Methoden

Prof. Dr.-Ing. E. Kirchner

Fachgebiet
Produktentwicklung
und Maschinenelemente

Institute for
Product Development
and Machine Elements

pmd

Prof. Dr.-Ing. Eckhard Kirchner

Otto-Berndt-Straße 2
64287 Darmstadt

Tel. +49 6151 16 - 21188
Fax +49 6151 16 - 21181
office@pmd.tu-darmstadt.de

Betreuer:
Moritz Schäfle, M.Sc.

Ausgegeben:
01. Dezember 2023

In-situ Tailoring of Functional Properties for the LPBF-Process of Pure Copper

Abstract

This thesis explores the application of the machine learning method physics-informed neural network (PINN) for the laser powder bed fusion (LPBF) process of pure copper. Given the process parameters, the accuracy of the predictions for relative densities of this method are compared to the traditional methods of a simple artificial neural network (ANN) and a linear regression. The results show the superiority of the PINN method in all tested scenarios, especially for complex and small data sets. Various data sets containing both literature data using red lasers and experimental data using a green laser are employed and show that a transfer of knowledge from red laser data to green laser data is feasible and beneficial. For red laser data from literature, the PINN method produces predictions with a mean squared error (MSE) of 4.58, as opposed to a MSE of 14.84 and 18.47 for a simple ANN and linear regression respectively. Using the transfer of knowledge for green laser data and reduced set of training data, PINN predictions exhibit a MSE of only 2.46, while a simple ANN and linear regression lead to a MSE of 7.29 and 5.19. These error values mean that for red laser data, the PINN method yields predictions that deviate 2.14 percentage points from the actual value and for green laser data the deviation is only 1.57 percentage points.

In addition, this thesis conducts a thorough literature review on the influence of process parameters on density, microstructure, and mechanical and electrical properties. This knowledge provides valuable insights for the further development of PINN for other outcomes, such as the microstructure and related functional properties. Furthermore, a test series for the LPBF process of copper using a green laser is presented for a laser spot diameter of 50 µm using an analytical approach focused on the geometry of the melting pools. This design of experiments serves as a first step for the systematic exploration of LPBF process parameters to achieve a broad analysis on their effects on density, microstructure, and functional properties.

This study pioneers in the application of PINN on the LPBF process of copper and through that, advances the research concerning the analysis of the in-situ influence through the modification of process parameters. The results help to further the LPBF process optimization in the face of lean process development with implications for the broader field of additive manufacturing.

TECHNISCHE UNIVERSITÄT DARMSTADT

pmd
produktentwicklung
maschinenelemente

Inhaltsverzeichnis

List of Figures

List of Tables

List of Appendix Figures, Tables, Codes

List of Abbreviations

AE	Autoencoder
ANN	Artificial Neural Network
CAD	Computer Aided Design
CFV	Compressed Feature Vector
DK	Domain Knowledge
DNN	Deep Neural Network
FFF	Fused Filament Fabrication
GRU	Gated Recurrent Unit
LED	Linear Energy Density
LHS	Latin Hypercube Sampling
LPBF	Laser Powder Bed Fusion
LSD	Laser Spot Diameter
LSTM	Long Short-Term Memory
MSE	Mean Square Error
NN	Neural Network
PDE	Partial Differential Equation
PIN	Physics-Informed Network
PINN	Physics-Informed Neural Network
pmd	Fachgebiet Produktentwicklung und Maschinenelemente
PSD	Powder Size Distribution
RFR	Random Forest Regression
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SDNN	Stochastic Data Generation with Neural Networks
SLM	Selective Laser Melting
TUD	Technische Universität Darmstadt
VED	Volumetric Energy Density

List of Symbols

A_a	Absorptivity [%]
$A_{a,c}$	Absorptivity (conduction) [%]
$A_{a,k}$	Absorptivity (keyhole) [%]
A_{el}	Elongation at break [%]
AR	Aspect ratio [−]
C_p	Heat capacity [$\frac{\mathrm{J}}{\mathrm{kg}*\mathrm{K}}$]
D	Powder size distribution [µm]
F	Test force [N]
G	Temperature gradient [$\frac{\mathrm{K}}{\mathrm{m}}$]
HV	Microhardness [HV]
LED	Linear energy density [$\frac{\mathrm{J}}{\mathrm{mm}}$]
P	Laser power [W]
P_e	Péclet number [−]
P_{max}	Maximum laser power [W]
R	Solidification rate [$\frac{\mathrm{K}}{\mathrm{s}}$]
R_e	Yield strength [MPa]
R_m	Tensile strength [MPa]
T	Temperature [K]
T_0	Substrate temperature [K]
T_m	Melting temperature [K]
T_v	Evaporation temperature [K]
VED	Volumetric Energy Density [$\frac{\mathrm{J}}{\mathrm{mm}^3}$]
W_m	Lowest laser power needed [W]
d	Laser spot diameter [µm]
d_e	Laser spot diameter (1/e value) [µm]
d_m	Melt pool depth [µm]
h	Hatch distance [µm]

n_e, m_e	Linear coefficients for Péclet number [−]
t	Layer thickness [µm]
v	Scanning speed [$\frac{\mathrm{mm}}{\mathrm{s}}$]
w_m	Melt pool width [µm]
η	Efficiency [%]
κ	Thermal conductivity [$\frac{\mathrm{W}}{\mathrm{m}*\mathrm{K}}$]
λ	Wavelength [nm]
ρ	Density [$\frac{\mathrm{kg}}{\mathrm{m}^3}$]
ρ_r	Relative density [%]
σ	Electrical conductivity [$\frac{\mathrm{S}}{\mathrm{m}}$]

1 Motivation

In recent years, there has been a notable increase in research regarding the use of copper within the laser powder bed fusion (LPBF) process. While many addressed copper alloys, pure copper has also sparked high interest due to its excellent thermal and electrical properties. Due to the geometric flexibility through the LPBF process, there are many more possibilities to effectively utilize these properties in fields such as aerospace. However, these studies have almost exclusively focused on the optimization of the relative density of copper by changing printing parameters through an excessive number of experiments. Some studies have also measured mechanical and electrical properties of the printed copper samples, few have examined the effect of heat treatment or other post processes on these properties.

There is a big research gap in the deliberate influence and control of these properties, as well as of the density. This thesis aims to address this research gap by providing insight on the interplay of process parameters with specific focus on machine learning techniques, more specifically Physics Informed Neural Networks (PINN). The primary goal is to modify and utilize this PINN method, so it can be applied to this specific printing process to predict the relative densities of copper samples given specific parameters.

This study serves as an initial step towards the ability to adapt the LPBF process to complex requirements, such as the intentional modification of functional properties. Through the ability to predict relative densities, future achievements may involve the targeted influence of the microstructure, such as grain sizes, shapes, and direction and thus the properties, while maintaining high density. Understanding this relationship provides information on the possibility of in-situ influence, making ex-situ treatments obsolite. In-situ tailoring during the process could additionally lead to the opportunity to produce parts that possess different properties in different areas, optimizing the usability and efficiency. This enables the production of parts with countless combinations of different properties only using pure copper while saving time, money, and material.

The application of the PINN method in this context has huge implications on lean process development, as it allows a substantial reduction in the number of experiments necessary to achieve the desired results. While this study specifically emphasizes on the application of this method for relative densities, it provides the necessary foundation for future pathways exploring mechanical and electrical properties.

By focusing on the application of the PINN method, this study does not only contribute meaningfully to the future of additive manufacturing but also offers a promising alternative towards a more advanced and sustainable development process. This is especially significant as presently, process development and component design are obtained iteratively through elaborate and resource-intensive repetitions of the process.

2 State of the Art

In the following chapter, the state of the art of relevant topics for this study are presented. This aims to provide a comprehensive overview of recent developments in the laser powder bed fusion (LPBF) process of copper. It explores literature concerning the microstructure, as well as mechanical and electrical properties of other functional materials to bring insight into the present research gap when it comes to copper. Lastly the foundation to understand the principles of physics-informed neural networks (PINN) is provided, as well as a detailed explanation of the method on which the subsequent research builds on.

2.1 Laser Powder Bed Fusion of Pure Copper

Laser powder bed fusion (LPBF), also called selective laser melting (SLM), is an additive manufacturing process, in which a laser goes through a metal powder bed, melting and fusing the powder together, based on the computer aided design (CAD) data provided. This powder bed is layered on a substrate and once the laser has gone over a layer, another layer is spread above it using a recoater, for instance a roller, brush or blade. This process is repeated until the desired object is produced.

Additive manufacturing methods in general propose great advantages compared to traditional manufacturing methods. These, among others, include high material efficiency, the production of complex components, production on demand and the use of fewer tooling requirements[1]. LPBF shows great advantages due to its high flexibility and fast production regarding complex geometries with low material waste and the possibility the customize products with low costs[2]. Key application areas for LPBF processed parts include the aerospace, automotive and medical field[3]. LPBF can completely melt the metal powder to produce components that possess high densities comparable to and mechanical properties superior to traditional manufacturing methods.

This high geometric flexibility can increase the electrical and thermal performance of components, as that is not only determined by material properties such as electrical and thermal conductivity, but also by the geometric design[4]. LPBF therefore shows great potential to not only increase the electrical and thermal performance from a microstructural way, but also from the geometry perspective – a way that is extremely limited in conventional manufacturing. However, the LPBF process is highly sensitive to process parameter changes and requires extensive experimentation to achieve the desired properties.

Pure copper is an established material that is often used due to its high thermal and electrical conductivity, making excellent heat exchangers and electrical devices[5]. As the manufacturing industry is rapidly evolving, higher requirements are set. Combined with complex geometries through LPBF, the

[1] Trevisan et al. (2017), p. 1.

[2] Yan et al. (2020), p. 2.

[3] Imai et al. (2020), p. 1.

[4] Colopi et al. (2019), p. 2473; Jadhav et al. (2021), p. 2.

[5] Ikeshoji et al. (2018), p. 396; Trevisan et al. (2017), p. 1.

potential for copper parts with profound conductivity efficiencies is very high[6]. However, regarding conventional LPBF, this process is primarily suited for materials that have low reflectivity and low thermal conductivity[7]. This means that pure copper is not an optimal material due to its contrary properties. Copper has an extremely low laser absorption rate near the infrared wavelength (1060 nm – 1080 nm) as seen in Figure 2.1, which is the laser used for conventional LPBF. It must be kept in mind that Figure 2.1 shows the absorptivity of bulk copper and not copper powder and can therefore only serve as a qualitative representation for copper powder. Copper's high reflectivity and high thermal conductivity reduce the energy that is available for the melting of the copper powder, limiting the maximum local temperature[8]. This results in defects in the finished product, with porous parts lowering the relative density. This leads to poor mechanical and electrical properties.

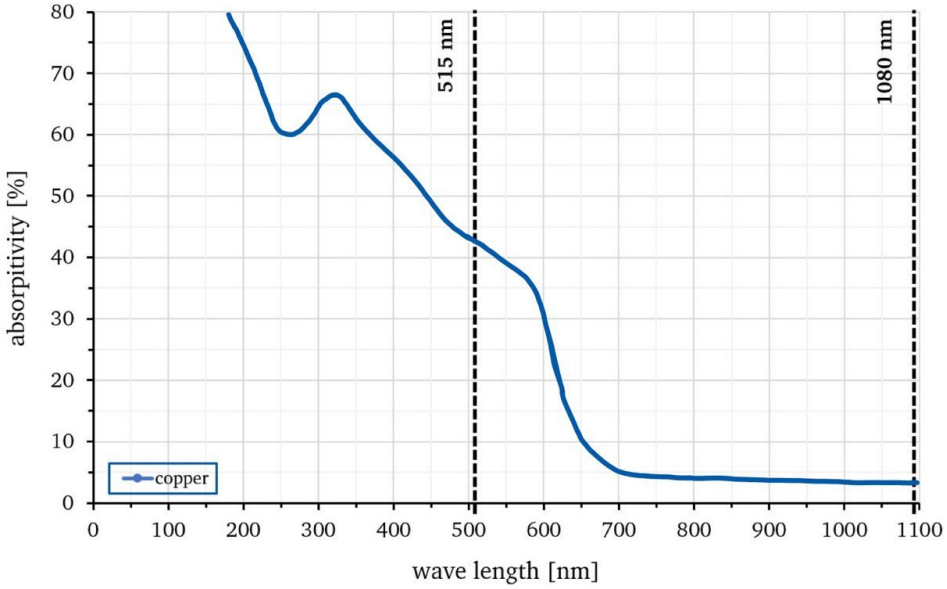


Figure 2.1: Absorptivity of copper at room temperature[9]

Because of this, many studies in the past have examined LPBF with mixed copper powders, as these alloys allow for better laser absorption and thus higher densities. Copper alloys are also associated with higher strengths. Due to pure copper's high impurity sensitivity, copper alloys have significantly reduced thermal and electrical conductivities[10]. Especially in electrical engineering, where the electrical and thermal conductivity of copper is pivotal, the use of pure copper is indispensable. This is especially, as copper is cheaper than silver, which is the only metal with a higher electrical conductivity[11]. One of the most important applications for pure copper are heat exchangers[12], which are both needed

[6] Jadhav et al. (2019), p. 2.

[7] Yan et al. (2020), p. 2.

[8] Bonesso et al. (2021), p. 256.

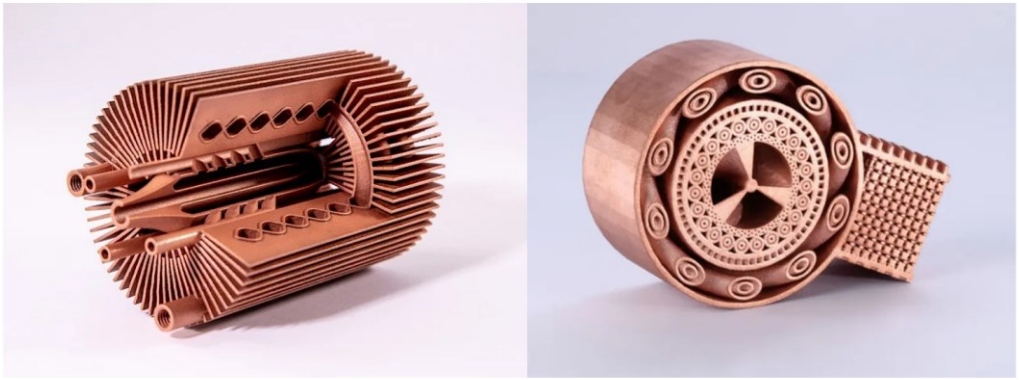
[9] Based on Hummel et al. (2021), p. 172.

[10] Ikeshoji et al. (2018), p. 396; Silbernagel et al. (2019), p. 2.

[11] Matula (1979), p. 1163 & p. 1262.

[12] Colopi et al. (2019), p. 2473; Imai et al. (2020), p. 1.

in automotive and aerospace manufacturing, among many other fields. Figure 2.2 shows two examples of copper heat exchangers that are fabricated using LPBF. Because of the flexibility regarding complex geometries, the individual specific requirements can be met while using the minimal amount of material, making copper parts especially useful for aerospace engineering, where lightweight components are essential[13]. Another application are copper coils used in electrical motors, which can be customized to achieve the optimum use of space, therefore reducing the size and weight of those[14].



a) b)

Figure 2.2: Two types of pure copper LPBF heat exchangers[15]

As it can be seen in Figure 2.1, the absorptivity of copper steeply increases with decreasing wavelength. Instead of using copper alloys, pure copper can be used while using a smaller-wavelength laser, such as a green laser with $\lambda \approx 515$ nm.

2.2 Effect of Process Parameters on Relative Density

An important aspect in the LPBF fabrication process of pure copper is the relative density of the manufactured parts. Many studies have attempted to optimize the relative density by modifying the parameters. Relevant parameters include the laser power P [W], laser scanning speed v $[\frac{\mathrm{mm}}{\mathrm{s}}]$, hatch distance h [µm], layer thickness t [µm], laser spot diameter (LSD) d [µm], and powder size distribution (PSD) D [µm]. A graphical representation can be seen in Figure 2.3. Furthermore, parameters regarding the exposure to the laser play a big role as well. This includes scanning strategy, the times the laser goes over an area, and potential waiting periods that influence thermal processes. These parameters can ultimately affect the electrical and mechanical properties of the finished product, namely thermal conductivity κ $[\frac{\mathrm{W}}{\mathrm{m}*\mathrm{K}}]$ and electrical conductivity σ $[\frac{\mathrm{S}}{\mathrm{m}}]$, tensile strength R_m [MPa], yield strength R_e [MPa], microhardness based on Vickers HV [HV] and the elongation at break A_{el} [%].

[13] Madonna et al. (2018), p. 7.

[14] Silbernagel et al. (2019), p. 1.

[15] a) *Https://delva.Fi/en/copper/*, b) *Https://de.Eos.Info/de/3d-Druck-Material/metalle/kupfer*.

The production of high-density parts is crucial for various reasons. For one, while not being the sole contributor, high density generally leads to better mechanical and electrical properties. Apart from that, the lack of pores also mean that the produced part is more consistent in its properties throughout, which is important for applications in which certain properties are pivotal for its use, resulting in higher efficiency.

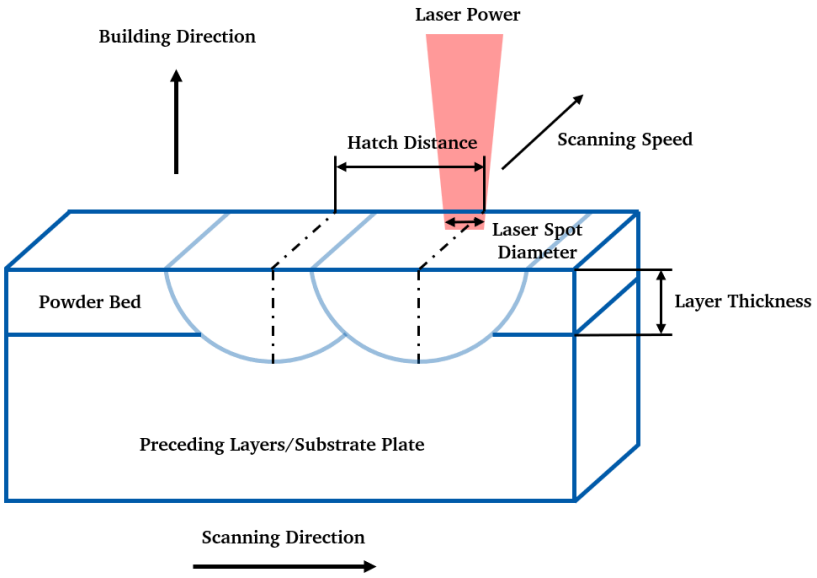


Figure 2.3: LPBF process parameters[16]

To understand and influence the microstructure and to achieve high densities, the laser melt pool is an important aspect to look at. A melt pool is formed when the laser irradiates the powder bed and the powder melts. Multiple melt pools in a row form melt tracks, which follow the direction of the laser. Laser melt pools can either absorb or scatter radiation, which leads to heat transfer and transformations. The melting behavior strongly depends on the parameters of the LPBF process. Through the high reflectivity and thermal conductivity of copper, laser energy is lost to the surroundings. This leads to defects such as partially melted powder or discontinuities in the melting track[17]. All that makes the use of copper difficult for LPBF. It is important to ensure a sufficient overlap of the melt pools in all three dimensions: in the scanning direction for continuous melt tracks, between melt tracks for dense layers and in the building direction, to minimize porosities between the layers.

Linear Energy Density: Laser Power and Scanning Speed

Reaching high density copper parts through LPBF has proven to be a rather complex task. Earlier studies with conventional machines using red laser and a laser power of around 200 W have led to low relative densities of under 90 %. There have been vastly different attempts to increase the relative density of LPBF fabricated copper through the modification of the printing parameters. The idea is to increase the absorptivity and localize the necessary energy to achieve sufficient melting and low porosities. Through these modifications relative densities of over 99 % were achieved, proving that the

[16] Based on Yap et al. (2015), p. 3.

[17] Guan et al. (2019), p. 1389.

LPBF process is valid for fabricating dense copper parts[18]. The most common is the modification of laser power, followed by scanning speed. While conventional LPBF machines use laser powers ranging from 100 W – 400 W, there have been high power LPBF systems with laser powers up to 1000 W. However, these high power lasers have shown to damage certain optical components in the LPBF printer[19].

Laser power and scanning speed are often looked at in pairs, as a higher laser power with a fixed scanning speeds leads to wider melting tracks, same as a slower scanning speed with fixed laser power[20]. If the scanning speed is too high or the laser power too low, the track width decreases, potentially leading to melting track discontinuities. To represent this relationship, the Linear Energy Density (LED) LED $[\frac{\mathrm{J}}{\mathrm{mm}}]$ is introduced:

$$LED = \frac{P}{v} \tag{1}$$

with P = laser power and v = scanning speed

If the LED falls below a certain value, the energy is insufficient and causes irregular melting tracks, disrupting the continuity. The low energy input causes poor fluidity of the melt pool, leading to unmelted powder and pores, called lack of fusion porosities. With increased LED, the fluidity increases, leading to lower porosity and thus a higher relative density. However, a too high LED leads to excess energy, which has negative impacts on the accuracy of the melting tracks. This excess energy leads to convective motion and the vaporization of the laser melt pool. This causes the surrounding gas to be trapped in the melt pool, resulting in circular pores inside the melt pool called keyhole porosities, again, decreasing the relative density[21]. Figure 2.4 shows a simplified version of the interplay between laser power and scanning speed for metals in general to achieve low porosity parts.

[18] Jadhav et al. (2021), p. 2.

[19] Jadhav et al. (2019), p. 16.

[20] Guan et al. (2019), p. 1391.

[21] Guan et al. (2019), p. 1391-1393, Yan et al. (2020), p. 14.

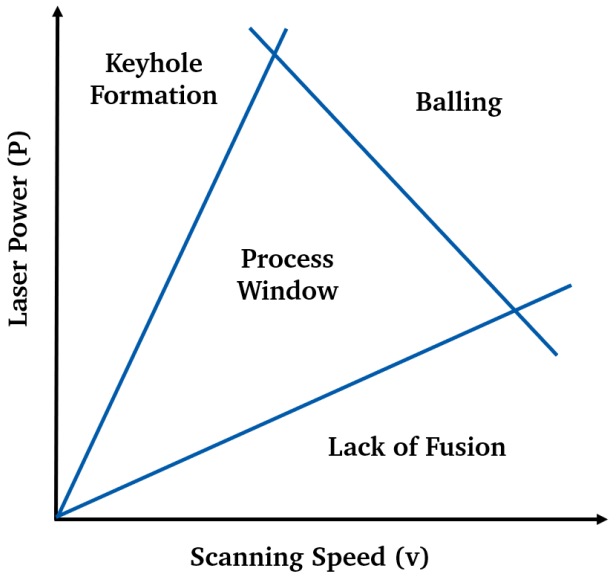


Figure 2.4: Influence of laser power and scan speed on defects[22]

Volumetric Energy Density: Hatch Distance and Layer Thickness

The principle of LED can be expanded into the Volumetric Energy Density (VED) VED $[\frac{\mathrm{J}}{\mathrm{mm}^3}]$. It is defined in Equation 2 as a function of laser power P, scanning speed v, hatch distance h, and layer thickness t:

$$VED = \frac{P}{v * h * t} \tag{2}$$

For the same reasons as LED, VED has an optimal range of values that minimizes defects. Imai et al.[23] found that the optimum VED for copper is much higher than for other, low conductivity materials such as titatium. This is due to copper's high reflectivity in the infrared wavelength and its high thermal conductivity which lead to narrower melt pools[24]. The effective VED VED_{eff} $[\frac{\mathrm{J}}{\mathrm{mm}^3}]$ describes how much of the emitted laser energy is absorbed and can be calculated as follows (for a qualitative representation of absorptivity A_a see Figure 2.1):

$$VED_{eff} = A_a * VED \tag{3}$$

However, solely looking at the energy density is not enough, as Jadhav et al.[25] have found that there is a minimum laser power that needs to be exceeded. Any laser power below that is not enough to completely fuse the copper powder particles regardless of the scanning speed. This leads to areas with

[22] Based on Calignano et al. (2019), p. 8

[23] Imai et al. (2020), p. 4.

[24] Qu et al. (2021), p. 2.

[25] Jadhav et al. (2021), p. 17-18.

unmolten powder and unstable melt tracks. This means a lower scanning speed was not able to compensate laser power that is too low, even with the same resulting energy density. Gargalis et al.[26] observed the relationship between laser power and absorptivity and found a steep increase in absorptivity at a certain laser power level. This suggests a transition between two modes, which will be explained in the subsequent sections. Similarly, they found that there is a certain scanning speed that cannot be surpassed as it will disrupt the stability of the melt pool. This will lead to discontinuous melt tracks and the so-called balling effect occurs, leading to rough surfaces. Contrary to what Figure 2.4 suggests, a lower laser power cannot compensate for that. Also, research has shown that the maximum relative density of LPBF pure copper occurs at a lower VED, when the laser power was higher as opposed to a high VED through low scanning speed, hatch distance, or layer thickness. Therefore, at least in the case of copper, only inspecting the VED is not sufficient in order to predict the relative density of the manufactured part[27]. Tang et al.[28] have also presented experimentally, that the same energy density can lead to different porosities with stainless steel. That means that the parameters also need to be considered individually. Nevertheless, VED can be used as a reference point, while keeping its shortcomings in mind.

Ikeshoji et al.[29] researched the importance of hatch distance and found that there is an optimal distance. If the hatch distance is too big, the laser melt pools will not have sufficient overlap, leaving lack of fusion voids, even if the melt tracks are stable and continuous. These voids will lead to porous parts, decreasing density. A smaller hatch distance would allow the melting down of these parts, increasing the density. The authors have shown in their experiments that if the hatch distance is too small, there might be an instability of the formation of laser melt pools, causing them not to form. This can be attributed to the high thermal conductivity of copper, leading to lack of fusion voids as well. Qu et al.[30] also found that both a too big and a too small hatch distance will lead to lack of fusion. This again proves that the VED does not provide enough insight, as a further decrease in hatch distance would simply lead to a higher VED, while the effect on density changes.

With smaller layer thickness, less laser power is needed for efficient melting as the melt pool depth, which is influenced by laser power[31], needed to achieve an overlap with the layer below, is smaller as well.

There are also parameters which are not considered in VED that influence the absorptivity. Instead of adjusting the laser through its wavelength or laser power, Bonesso et al.[32] have used the approach of modifying the powder size distribution (PSD) of the used copper powder. They compared three different lots of copper powder and found that the PSD has an influence on relative density, surface roughness and mechanical, as well as thermal properties. They conclude that the finer the powder, the easier

[26] Gargalis et al. (2021), p. 2.

[27] Jadhav et al. (2021), p. 7.

[28] Tang et al. (2017), p. 8.

[29] Ikeshoji et al. (2018), p. 397.

[30] Qu et al. (2021), p. 10.

[31] Gargalis et al. (2021), p. 6; Jadhav et al. (2021), p. 11.

[32] Bonesso et al. (2021), p. 257-260.

it is to melt it. This is due to the higher surface area finer powder has, meaning that there is more surface area to absorb laser energy.

Another factor is the laser spot diameter (LSD). The laser spot usually has a Gaussian energy distribution, therefore the intensity in the middle of the molten tracks is higher than on the edges[33]. This is illustrated in Figure 2.5. With lower LSD the high intensity of the laser can be used more precisely, reducing the laser power needed[34].

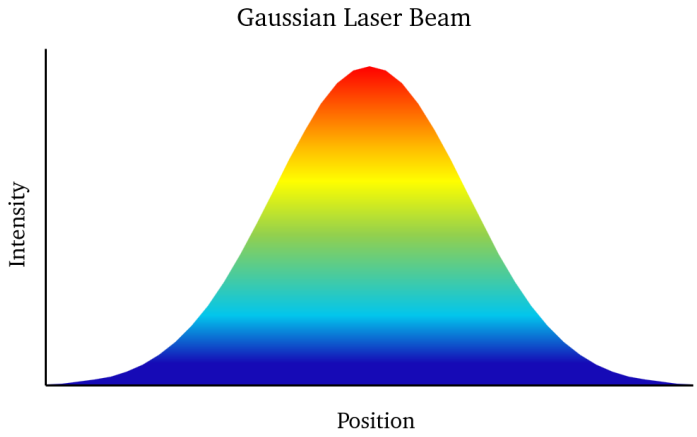


Figure 2.5: Gaussian Laser Beam

Melting Modes and Melt Pool Geometry

Laser melt pools can appear in two different melting modes: conduction mode and keyhole mode melting as shown in Figure 2.6. In conduction mode, a relatively shallow and wide melt pool is formed, and the heat transfer predominately happens through conduction. On the other hand, the keyhole mode is characterized through heat convection[35]. If the energy density is too low, it leads to conduction mode melting and transforms to keyhole mode melting, once the energy density is high enough. As established, this is not always the case with copper. If the minimum laser power needed is not achieved, the keyhole melting mode will not be achieved regardless of the energy density. This means that energy density is not a reliable unit. The required energy density strongly depends on which individual parameters are changed. This is why looking at the melting modes and melt pool geometries gives a more precise understanding of the influences of different printing parameters on the microstructure and resulting properties of the processed copper part.

[33] Guan et al. (2019), p. 1393.

[34] Qu et al. (2021), p. 11.

[35] Patel and Vlasea (2020), p. 1.

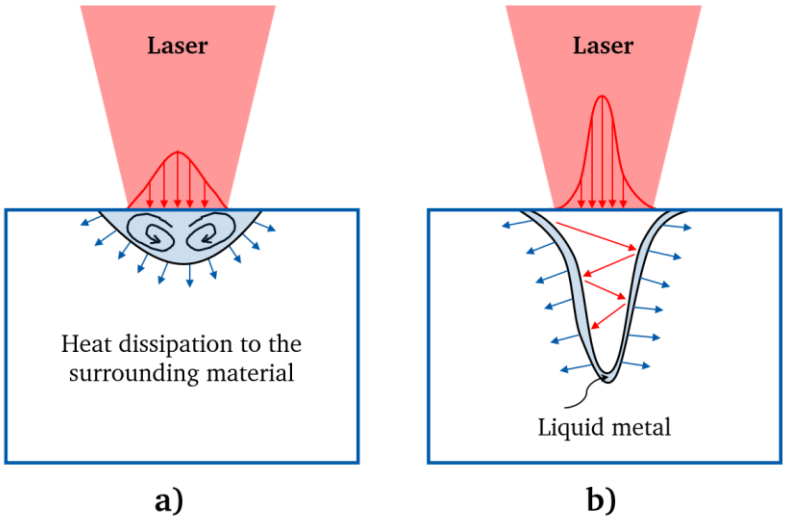


Figure 2.6: Melting modes: a) conduction mode, b) keyhole mode[36]

Jadhav et al.[37] particularly researched the transition between conduction mode and keyhole mode, both analytically and experimentally. To predict when this transition will take place, they proposed an aspect ratio, where the threshold value between these two modes is at 1 and backed it up experimentally. This so-called melt pool aspect ratio AR $[-]$ is defined in Equation 4 as the quotient between melt pool depth d_m $[\mu m]$ and melt pool width w_m $[\mu m]$:

$$AR = \frac{d_m}{w_m} \tag{4}$$

In conduction mode, there is a lower absorptivity because of the geometry of the conduction melt pool. Without the keyhole shape, the laser light cannot enter the melt pool deeply and therefore does not allow the light to reflect within the melt pool[38]. This also explains the aforementioned steep increase in absorptivity during the transformation to the keyhole mode, as the laser light can enter and be reflected by the melt pool walls multiple times before getting absorbed. Qu et al.[39] have shown that in conduction mode, the same or even higher energy density will lead to narrower melt pool widths than in keyhole mode.

In accordance with the melting modes, lack of fusion and keyhole porosities appear as follows: Lack of fusion porosities occur in the conduction mode, meaning if the aspect ratio is smaller than 1. This is the case if the absorbed laser energy is too low. Keyhole porosities occur in the keyhole mode if laser power is too high or scanning speed too low.

[36] Based on Jadhav et al. (2021), p. 21.

[37] Jadhav et al. (2021), p. 5-7.

[38] Jadhav et al. (2021), p. 23.

[39] Qu et al. (2021), p. 5.

In Qu et al.'s[40] study, the microstructure of thin wall samples during lack of fusion, high density and keyhole condition are compared. These three conditions can be equated to Jadhav et al.'s[41] convection mode, the stable keyhole mode and the unstable keyhole mode. Looking at the top surface, a branch like structure from the solidified melt pools is evident in the lack of fusion sample. The high-density sample showed stable and continuous melt tracks, while the keyhole sample showed wider tracks with extra powder sintered to the side. Regarding the side surface, the microstructure of the lack of fusion and keyhole sample was in accordance with previous studies with voids and unmolten powder, and keyhole pores respectively.

With fixed scanning speed, a higher laser power leads to a higher melt pool depth[42]. There are discrepancies in research regarding the influence of scanning speed with fixed laser power. While Jadhav et al. found that a lower scanning speed leads to a higher melt pool depth as well, Gargalis et al. could not observe any changes in melt pool depth with varying scanning speed. For melt pool width, Jadhav et al. found for different scanning speeds that laser power does not have an effect, while a decrease in scanning speed leads to an increase in melt pool width. However, this finding is contrary to Guan et al.[43], who found wider melt tracks with increased laser power. Another study showed that the width of the melt tracks increases with increasing energy density, but is ultimately dependent on the laser spot diameter[44]. The inverse effect of the scanning speed on the width of the melt pool has also been documented by Gargalis et al.[45], who investigated single melt tracks at different laser power and scanning speeds. For higher scanning speeds, an increase from 400 W to 500 W laser power did not influence the melt pool width. This suggests that scanning speed and laser power might only influence the depth and the width, respectively, under certain conditions.

Tang et al.[46] have presented a geometry-based approach to calculate the expected porosity through lack of fusion. The goal of this analytical method is to achieve predictions based on the process parameters, without having to perform the whole LPBF process. The idea behind their calculations is the same idea as presented at the beginning of this chapter: To avoid lack of fusion porosity, there needs to be a sufficient overlap between melt pools in all dimensions. To achieve full melting, the authors proposed:

$$\left(\frac{h}{w_m}\right)^2 + \left(\frac{t}{d_m}\right)^2 \leq 1 \tag{5}$$

with h = hatch distance, t = layer thickness, w_m = melt pool width, d_m = melt pool depth

[40] Qu et al. (2021), p. 4.

[41] Jadhav et al. (2021), p. 10.

[42] Gargalis et al. (2021), p. 6; Jadhav et al. (2021), p. 11. These also apply to subsequent mentions of the authors.

[43] Guan et al. (2019), p. 1391.

[44] Yan et al. (2020), p. 17.

[45] Gargalis et al. (2021), p. 4.

[46] Tang et al. (2017), p. 3.

Equation 5 must be fulfilled, otherwise there will be lack of fusion porosities. It is important to mention that this approach solely focuses on lack of fusion and does not consider any other porosities, such as keyhole porosities or even lack of fusion pores due to irregularities in the powder bed or in the process. Furthermore, Ikeshoji et al.[47] has shown that a too small hatch distance also leads to lack of fusion porosities. This is not accounted for here either. Even though the Rosenthal equation poses some flaws, it has been found accurate to experimental measures regarding melt pool geometry[48]. This is under regular conditions, meaning that there are no extremes in process parameters. The melt pool width w_m and melt pool depth d_m used in the equation are analytically estimated from the Rosenthal equation[49]. Through differentiation and approximation of that equation, Gordon et al.[50] proposed:

$$d_m = \sqrt{\frac{2 * P * A_a}{\pi * e * \rho * C_p * v * (T_m - T_0)}} \tag{6}$$

with P = laser power, A_a = absorptivity, ρ = density, C_p = heat capacity, v = scanning speed, T_m = melting temperature, T_0 = substrate temperature

The melt pool depth d_m is then assumed to be half of the melt pool width w_m:

$$w_m = 2 * d_m \tag{7}$$

These approximations using these formulas can only be calculated for conduction mode melting and not keyhole mode. The assumption of a semi-circular melt pool with double the width of the depth results in a static aspect ratio AR of 0.5, while Jadhav et al.[51] proposed that the conduction mode remains up to an aspect ratio of just below 1. So these assumptions are only true for a specific process window within the conduction mode. Another factor is that the validity of these equations may vary when it comes to materials with high thermal conductivity such as copper[52].

Using these approximations, an experimental investigation examined the general trend of lack of fusion numbers. The results showed that for pure copper, the threshold value for achieving high densities is closer to 0.5 rather than 1, as proposed. For lack of fusion numbers higher than 0.5, distinct lack of fusion porosities were observed. Overall, the lack of fusion number could not serve as a quantitative predictor of densities but could serve as a qualitative reference point, even though results have shown some outliers. This is a point to be further investigated. Furthermore, the results indicated that within this threshold, excessively low lack of fusion numbers might also lead to porosities. This may be attributed to the fact that this approach does not take keyhole mode melting and the associated keyhole porosities into account.

[47] Ikeshoji et al. (2018), p. 397.

[48] Tang et al. (2017), p. 6.

[49] Rosenthal (1941).

[50] Gordon et al. (2020), p. 4-5.

[51] Jadhav et al. (2021), p. 10.

[52] Gordon et al. (2020), p. 5.

Another point to be considered is that these approximations assume the influence of laser power and scanning speed on both melt pool depth and consequently melt pool width. As research has shown, this must not necessarily be the case. This again, shows the issue with the static assumption that the melt pool width is double the melt pool depth and therefore a direct proportional interdependence between them.

While using a red laser, Jadhav et al.[53] concluded that to reach the necessary effective laser absorption for low porosities, the keyhole mode must be achieved, but with the VED still low enough so that the keyhole porosities do not form yet. That was the case with an aspect ratio of 1.5. With a higher melt pool depth to width ratio, keyhole porosities form. These can occur because of the high thermal conductivity and reflectivity of copper, which rapidly dissipates the heat. That means in the solidification process, the time span is shorter for filling and closing the keyhole with liquid copper, increasing the possibility of trapping vapor bubbles which become the keyhole porosities. For a lower ratio the VED is not sufficient for efficient melting, resulting in lack of fusion porosities.

The Use of Green Lasers

Because pure copper has a higher absorptivity for shorter wavelengths, there is the possibility that when using green lasers, the effective laser absorption needed to achieve the full melting of the copper powder could be reached in the conduction mode, without needing the geometry of the keyhole mode to achieve sufficient laser absorption. This means extremely high relative densities could potentially be achieved in both keyhole mode and conduction mode. This allows for a much bigger processing window, leaving room for more process parameter combinations to achieve high densities in both modes[54]. Further studies must be conducted to confirm this.

Nordet et al.[55] concluded that the use of green lasers result in more stable keyhole formations. They also found an absorptivity of $A_a = 75\,\%$ of the copper powder, which decreased to $A_{a,c} = 50\,\%$ in the liquid state, meaning during conduction mode, and increased to $A_{a,k} = 80 - 90\,\%$ in the keyhole mode. This shows significant increases in all states, when compared to infrared lasers. For Cu-ETP powder, Gruber et al.[56] measured an increase in absorptivity of around 45 percentage points, from 31.99 % to 76.93 %, when switching from an infrared laser at 1064 nm to a green laser with a wavelength of 515 nm. As green lasers cause higher absorption, using lower laser power is expected to reach the same results as higher laser power with red lasers. This would solve the issue of damaging optical components that came with very high laser power[57]. While the use of blue diodes lasers at around 450 nm on copper have shown to result in even higher absorptivities, there are limitations regarding the focused beam diameter, as well as brightness which makes them

[53] Jadhav et al. (2021), p. 10.

[54] Jadhav et al. (2021), p. 20.

[55] Nordet et al. (2022), p. 10.

[56] Gruber et al. (2021), p. 6.

[57] Jadhav et al. (2019), p. 16.

less suitable for the LPBF process[58]. Therefore, green lasers serve as the best opportunity for stable and efficient melt pool formation to create highly dense copper parts.

2.3 Microstructure and Functional Properties

While the relative density of the copper samples influences their mechanical and electrical properties, as a higher density generally means less pores and therefore less defects, it is not sufficient to predict these. While retaining the same density, a sample can have vastly different microstructures consisting of grain size, shape, and direction among other factors. The microstructure determines the different mechanical and electrical properties. The process parameters can influence its development, as well as the thermal processes that ultimately determine the microstructure. This refers to heating and cooling rates, temperature gradients, solidification rates, undercooling and others[59]. Because studies on the effect of thermal processes in LPBF of pure copper are extremely scarce, studies investigating other metals and their alloys are also consulted in this section.

Thermal Processes

The interaction time between the laser and the powder plays a pivotal role. From that and the laser power and scanning speed, the temperature gradient G $[\frac{\mathrm{K}}{\mathrm{m}}]$, solidification rate R $[\frac{\mathrm{K}}{\mathrm{s}}]$, and cooling rate can be derived[60]. The temperature gradient describes the rate at which temperature changes over a distance and the solidification rate is the rate at which the material changes from its molten state to a solid state. Copper in general has a high solidification rate compared to other materials due to its high melting point and high thermal conductivity[61]. Research has shown that the ratio between these factors influence the shape of the grains as seen in Figure 2.7. The solidification rate is connected to the cooling rate as a faster cooling rate leads to a higher solidification rate, albeit the relationship is not necessarily linear. The LPBF process possesses high cooling rates compared to other manufacturing methods, as the melt pools in general are extremely small[62].

[58] Nordet et al. (2022), p. 2.

[59] Chowdhury et al. (2022), p. 2137.

[60] Chowdhury et al. (2022), p. 2137.

[61] Yan et al. (2020), p. 23.

[62] Karlsson et al. (2020), p. 1.

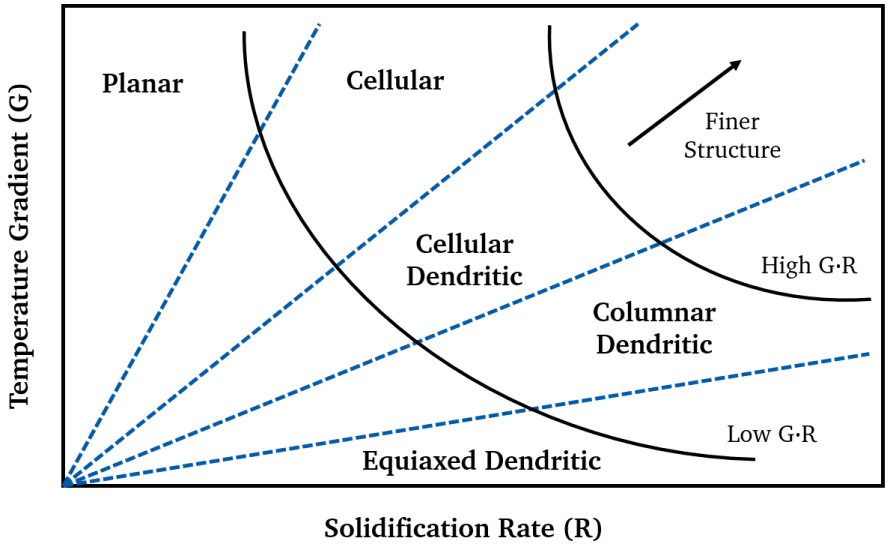


Figure 2.7: Influence of temperature gradient and solidification rate on grain shape[63]

Many studies have attempted to determine the relationship between process parameters and thermal processes in LPBF to understand and control the formation of the microstructure. Just like studying the density, only looking at the energy input such as the VED is not enough to determine the cooling rate, as it cannot predict the melt pool behavior, which is determinative of the microstructure[64]. It is also possible that the microstructure changes throughout the LPBF process, as the bottom has a higher cooling rate than the top of the sample. This is because during the process, heat is collected, thus leading to higher temperatures at the top[65], which is known as the heat accumulation phenomenon. The maximum temperature and melt pool depth, length, and width have shown to be increased in the second layer compared to the first[66]. The in-situ investigation and measurement of temperature distribution in the melt pools is difficult, so the experimental research on that is extremely limited. Hooper[67] was able to develop a method to measure the temperature during the process and found that the cooling rate had a maximum variation in different areas of the sample by 40-fold. Using a serpentine scanning strategy, he found that the beginning and end of each scanning line had the lowest cooling rate, as that was the area in which the laser turned, thus staying longer on that spot. This shows that the LPBF printed samples could possess different microstructures throughout the geometry. Further studies must be conducted.

Li and Gu used simulation of an aluminum alloy to determine the influence of laser power and scanning speed on the thermal processes. They found that the cooling rate increases slightly with increasing

[63] Based on Lippold (2015), p. 21.

[64] Chowdhury et al. (2022), p. 2137.

[65] Dai et al. (2017), p. 11.

[66] Li and Gu (2014), p. 859.

[67] Hooper (2018), p. 559.

laser power and significantly with increasing scanning speed. Samy et al.[68] on the other hand found a decrease in cooling rate with higher laser power due to the higher heat accumulation that comes with it. In any case, this means using a high laser power and low scanning speed as many studies suggested to achieve high densities, leads to rather slow cooling rates. This has a damaging effect on the mechanical properties of the processed part. This can be explained by the influence of the cooling rate on the grain size. With slower cooling rates, the grain size increases as it allows more time for the grain to grow[69]. However, mechanical properties such as strength or hardness favor small grain sizes as will be explained in subsequent sections.

Because the LPBF laser has a Gaussian energy distribution, the melt pool does not have a uniform energy distribution either, which affects the temperature gradient within that melt pool. In Hooper's experimental study and Li and Gu's simulation, a Gaussian temperature distribution for single laser pulses in the melt pool were found. The former found an approximately three times higher temperature in the middle of the melt pool than at the edge. The laser power has shown to have a considerable impact on the temperature gradient as the increase of laser power results in a linear increase in the temperature gradient. The effect of scanning speed was small, where a faster scanning speed only decreased the temperature gradient slightly[70]. The reason for that is that the laser power is directly connected to the temperature, while the scanning speed is connected to the temperature through the interaction time between laser and powder.

To summarize, the scanning speed is mainly responsible for the cooling rate and the laser power is mainly responsible for the temperature gradient. As cooling rate and solidification rate have a positive relationship and because the ratio between solidification rate and temperature gradient determines the solidification morphology, this suggests that the shape of the grains formed can theoretically be controlled through these process parameters.

Melt Pool Geometry on Microstructure

Jadhav et al.[71] examined the microstructure of LPBF pure copper with results that confirm the impact of laser power and scanning speed. When investigating the top layer of the sample, they found that using high LED through high laser power and low scanning speed resulted in deep melt pools and mainly planar solidification morphology and partly cellular solidification morphology were formed. As seen on Figure 2.7, this is attributed to a high temperature gradient and relatively low solidification rate. The high temperature gradient could be due to the high laser power used and the low solidification rate because of the low scanning speed. For lower energy densities, meaning lower laser power and higher scanning speed, the melt pools were less deep and more elliptical, with the top layer microstructure consisting of mostly cellular morphology. This means that the temperature gradient was

[68] Samy et al. (2023), p. 5.

[69] Gu et al. (2018), p. 18.

[70] Li and Gu (2014), p. 860-861.

[71] Jadhav et al. (2019), p. 8-9.

relatively lower and the solidification rate higher. This could be caused by the decrease in laser power and increase in scanning speed.

An important point to consider is that depending on the process parameters, the melt pool depth might considerably exceed the layer thickness. This means that the powder of one layer is melted multiple times, when the laser goes through the following layers. Therefore, the final layer differentiates itself from the inner layers, as it is the only one that is not remelted. This means the ratio between melt pool depth and layer thickness influences the microstructure as well as it determines how much of the layer below is remelted. Similarily, Jadhav et al.[72] did not discover any distinct structures in the middle layers as they did in the top layer. Instead, the middle layers showed rather relatively random crystal orientations. That can be explained by the successive remelting of those layers and the changed direction of the temperature gradient between the layers through the changing scanning direction based on the scanning strategy. Using a scanning strategy in which the scanning direction is kept the same, could lead to a more uniform remelting and thus a less random and more controlled middle layer morphology.

The size and shape of the melt pool is determinative for the microstructure and can be influenced by the process parameters as described in chapter 2.2. In keyhole mode, melt pools appear deeper when looked at vertically to the scanning direction and have an ellipse shape when looked at from above. Conduction mode melt pools are shallower and have a conical shape[73]. The melt pool size is associated with the grain size as bigger melt pools have lower cooling rates, leading to bigger grains[74]. The melt pool geometry also determines the direction of the biggest temperature gradient. Jadhav et al.[75] found a rotation of 10-45° of the maximal temperature gradient when comparing deep melt pools to elliptical melt pools. This means that the direction of the grain growth is also rotated. In general, the grain growth direction is dependent on the direction of the largest temperature gradient[76]. This can lead to anisotropy regarding the functional properties. The scanning strategy also has an influence on the grain growth direction. Two often used examples are illustrated in Figure 2.8. If grain growth occurs at an angle to the building direction, serpentine scanning will lead to a crosswise pattern throughout the layers, while unidirectional scanning will lead to uniform growth directions.

[72] Jadhav et al. (2019), p. 12-13.

[73] Jadhav et al. (2019), p. 19.

[74] Chowdhury et al. (2022), p. 2137.

[75] Jadhav et al. (2019), p. 19.

[76] Gu et al. (2018), p. 18.

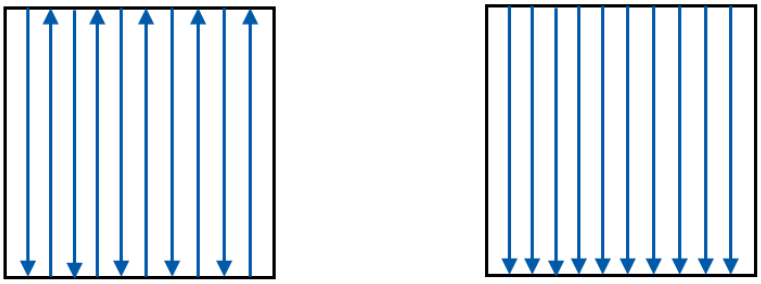


Figure 2.8: Scanning stategies a) serpentine, b) uniform

Mechanical Properties

A common way to modify the functional properties of copper and other metals is through ex-situ heat treatments. While the aim of this thesis is to provide the necessary information to achieve the in-situ tailoring of these properties, the studies on heat treatment can still provide information on the relationship between microstructure and properties. The longer the heat treatments and the higher the temperatures, the larger the grain sizes become[77]. For example, heat treatment has shown to decrease yield strength and tensile strength for LPBF copper samples, as well as microhardness, but to increase elongation at break[78]. Studies have also revealed that heat treatment increases electrical conductivity[79]. Table 2.1 depicts the specific property values of an as-processed and a heat-treated sample fabricated using the same process parameters. In the following, the relationship between microstructure and functional properties will be assessed through literature, to make suggestions on desirable microstructure for LPBF copper parts, depending on the application.

Table 2.1: Comparison as-processed and heat-treated samples[80]

Sample	**Yield Strength** [MPa]	**Tensile Strength** [MPa]	**Microhardness** [HV]	**Elongation at Break** [%]
As-processed	187±5.3	248±8.5	84±4.2	9.2±2.12
Heat-treated	51±8.2	215±7.2	73±5.2	30±3.04

[77] Qu et al. (2021), p. 8.

[78] Qu et al. (2021), p. 8; Yan et al. (2020), p. 20-21.

[79] Silbernagel et al. (2019), p. 13.

[80] Taken from Yan et al. (2020), p. 20-21.

The Hall-Petch relationship describes the relationship between grain size and yield strength of metals and their alloys, in which smaller grain sizes lead to higher strength[81]. The reason for this is that the smaller the grain size is, the higher the grain boundary density becomes. Dislocations are irregularities within the individual grains and allow for movements that can cause deformation as shown in Figure 2.9. These grain boundaries obstruct the dislocation movement, meaning that more stress must be applied to reach the threshold at which plastic deformation takes place. This is called grain boundary strengthening. Yan et al.[82] investigated LPBF copper samples and established that both yield strength and tensile strength were improved with smaller grains and low porosities. For the latter, high density has proven to be crucial. With lack of fusion porosities, the voids were observed to be formed in regions between laser tracks, which facilitate the stress concentration and crack formation locally, leading to earlier fracture. Because in the high-density samples, porosities are very low, the stress concentration is more likely to form around the melt pool front where grains have formed, resulting in higher tensile strength. The most important factors for high strength are small grains and low porosities. It should be mentioned that the direction in which the sample is printed, makes a difference here. Gruber et al.[83] investigated the microstructure of LPBF pure copper and found columnar grains that grow parallel to the building direction. Because of the shape of the grains and the layering nature of the LPBF process, the sample showed anisotropy regarding tensile strength. This means the building direction is another factor to consider when fabricating parts. Samples have shown to have higher tensile strength in the direction of the scanning, meaning the direction perpendicular to the building direction[84]. Based on the desired functionality of the part, the corresponding preparations regarding printing direction can be made.

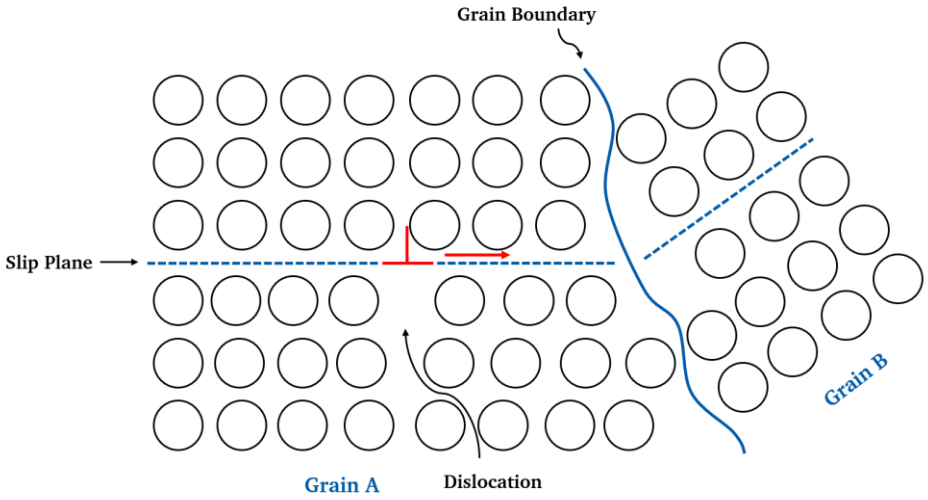


Figure 2.9: Dislocation movement on grain boundary[85]

[81] Cordero et al. (2016).

[82] Yan et al. (2020), p. 1.

[83] Gruber et al. (2021), p. 7.

[84] Chowdhury et al. (2022), p. 2143-2144; Gruber et al. (2021), p. 8; Karlsson et al. (2020), p. 8.

[85] Based on Callister and Rethwisch (2014), p. 230.

Microhardness refers to the hardness of a material using small-scale indenters and low testing loads. Yan et al.[86] found that the fewer microstructural defects and therefore higher density the investigated copper sample had, the higher the microhardness was. Other studies have determined that some defects such as residual stress can increase hardness when managed correctly[87]. This suggests that high density does not necessarily provide the best hardness. However, managing that is extremely complex, which is why in general, high densities are still desirable. For nickel-based parts, Gu et al.[88] found variations for microhardness depending on the grain shape. There, columnar dendrites growing in uniform direction had the smallest microhardness, while fine cellular dendrites or equiaxed grains had the highest microhardness. Furthermore, fine grain structures through fast cooling rates enhance microhardness[89]. Furukawa et al.[90] have experimentally confirmed that the Hall-Petch relationship applies to microhardness as well. Just as yield and tensile strength, small grains and low porosities are favorable. Unlike tensile strength, microhardness is mostly independent of the building direction[91].

Elongation at break determines the difference between a brittle and a ductile material and describes the percentage of how much a sample can be stretched before breaking. Ductility is often associated to have an inverse relationship with hardness and strength[92]. However, this is not necessarily true. Indeed, literature has shown that heat treatment, which increases the grain sizes[93], has led to significantly improved ductility[94]. It is important to keep in mind that heat treatment does not only increase grain sizes but also reduces internal stress, which would otherwise promote cracks and fracture[95]. Therefore, the improved ductility cannot definitively be contributed to the grain sizes but rather to other factors, suggesting that as-printed samples with big grain sizes might not be as ductile. Instead, studies have shown that smaller grains favor elongation at break. Through grain refinement, ductility can be increased[96], while strength and hardness are increased as well. This is because with increasing grain boundary density, the structure is more uniform, hindering the formation of cracks through plastic deformation[97]. In the end, elongation at break can be understood as the amount of dislocation movement possible before failure. While more dislocation movement per grain is possible with bigger grains, smaller grains help to prevent cracks and thus early fracture, resulting in a higher total

[86] Yan et al. (2020), p. 19.

[87] Chowdhury et al. (2022), p. 2144.

[88] Gu et al. (2018), p. 10.

[89] Dai et al. (2017), p. 18; Yan et al. (2020), p. 19.

[90] Furukawa et al. (1996).

[91] Chowdhury et al. (2022), p. 2144.

[92] Chowdhury et al. (2022), p. 2145.

[93] Qu et al. (2021), p. 8.

[94] Yan et al. (2020), p. 20.

[95] Qiu et al. (2014), p. 1963.

[96] Gu et al. (2015), p. 28; Hosseini et al. (2013), p. 8.

[97] Gu et al. (2015), p. 28.

dislocation movement. Furthermore, ductility is also dependent on density, with higher density leading to higher ductility[98].

To conclude, previous research has shown that the desired microstructure for good mechanical properties consists of high densities and small grain sizes while taking the building direction and grain shapes into consideration.

Electrical and Thermal Properties

Investigating the electrical conductivity of as-printed pure copper samples, Qu et al.[99] have determined a positive relationship with density. In high density samples, electrical conductivity was higher than in lack of fusion or keyhole porosity samples. In general, conductivity and mechanical strength have an inverse relationship. Attempts to increase mechanical properties usually results in the decrease of thermal and electrical properties[100]. This can be attributed to the dependency of electrical conductivity on the grain boundaries and dislocations of the crystal structure. An uneven distribution of the grains in the microstructure can impair electrical conductivity[101]. Qu et al.[102] suspect that high cooling rates which lead to small grains and a higher dislocation energy might be the reason for poor electrical conductivity. This causes the conduction electrons to scatter by grain boundaries or dislocations, which increases the electrical resistivity, thus diminishing electrical conductivity[103]. So at the same relative density, higher laser power leads to higher conductivity through the higher temperature and slower cooling rate that come along with it[104].

Thermal conductivity is frequently calculated through the electrical conductivity using the Wiedemann-Franz law[105] and therefore follows the same criteria as electrical conductivity. Bigger grain sizes are more desirable for parts that require high conductivity.

In conclusion, preceding studies have shown that to achieve good mechanical and electrical properties, the process parameters must be adjusted in a way that maintains high density while changing the thermal processes that determine the microstructure. Mechanical properties favor smaller grain sizes while electrical properties are improved with bigger grain sizes. Thereby, it seems that the mechanical properties come at the expense of the electrical properties, as depicted in Figure 2.10.

[98] Chowdhury et al. (2022), p. 2145; Jadhav et al. (2021), p. 14.

[99] Qu et al. (2021), p. 7.

[100] L. Lu et al. (2004), p. 1.

[101] Qu et al. (2021), p. 11.

[102] Qu et al. (2021), p.11.

[103] L. Lu et al. (2004), p. 1; Qu et al. (2021), p. 10-11.

[104] Qu et al. (2021), p. 11.

[105] Qu et al. (2021), p. 8.

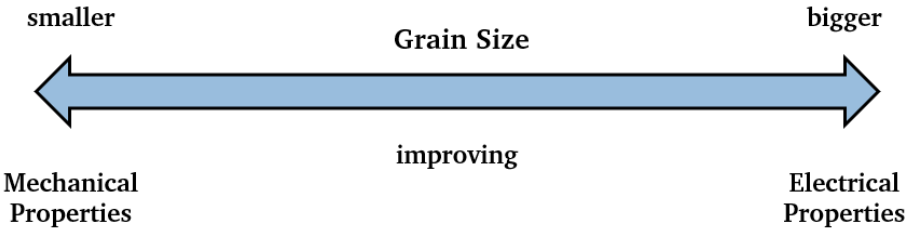


Figure 2.10: Dependency of functional properties on grain size

2.4 Physics Informed Neural Networks

The fabrication of LPBF copper parts with specific mechanical and electrical properties is dependent on the process parameters and their complex interrelationships. Most research has taken the trial-and-error approach, in which the printing process is repeated many times using different combinations of process parameters until the desired properties are reached. However, this costs time and money and the parameters might even vary with different designs[106]. This is why using machine learning techniques can greatly support the target-oriented printing process and increase the reliability of predictions for different components.

Physics-informed neural networks (PINN) are a method of machine learning that differs from conventional methods, as it not only considers data sets but also the existing knowledge, such as physical laws[107]. It does not derive the solution purely based on data, making it more robust, especially in the engineering context. In the conventional method of machine learning extremely large amounts of training data are used for the Neural Network (NN). A common issue in the research of NN is the lack of generalization, making it difficult to simulate more complex models[108]. For any inputs outside of the training distribution, the results will not be very reliable. The approach of simply further increasing the training data is no solution, as it is impossible to collect all data sets for every existing possibility. In many practical settings, either only limited data are available, or the experiment observations are noisy. Instead, PINN utilize the already existing physical knowledge, which provides further constraints that the PINN must satisfy[109]. This gives the NN more of a scientific understanding, eliminating predictions that are physically impossible, thus reducing the possible solutions to reasonable amount. These laws and principles of physics are used through mathematical models, more precisely partial differential equations (PDE)[110]. This reduces the data sets needed to train the NN immensely, as it also increases the informational value of the available data[111]. The more physical knowledge is included, the

[106] Kapusuzoglu and Mahadevan (2020), p. 1.

[107] Raissi et al. (2019), p. 686.

[108] Moseley et al. (2020), p. 1527.

[109] Raissi et al. (2019), p. 687.

[110] Cuomo et al. (2022), p. 2.

[111] Raissi et al. (2019), p. 687.

less data are needed. This is especially important in additive manufacturing such as LPBF, as it is difficult to carry out large scale experiments due to the high costs and time consumption.

PINN can be seen as an extension of NN. Cuomo et al.[112] describe PINN in three sections: a Neural Network (NN), a Physics-Informed Network (PIN), and a feedback mechanism. NN or rather artificial NN (ANN) are machine learning techniques that simulate biological organisms through neurons. These neurons can receive inputs and send out outputs. Multiple neurons make up a layer. The basic structure of an ANN consists of an input layer, at least one hidden layer, and an output layer as seen in Figure 2.11. In general, ANN can be differentiated between shallow ANN and deep NN (DNN). The difference is that while shallow ANN only have one hidden layer, DNN have multiple, allowing them to model more complicated relationships[113].

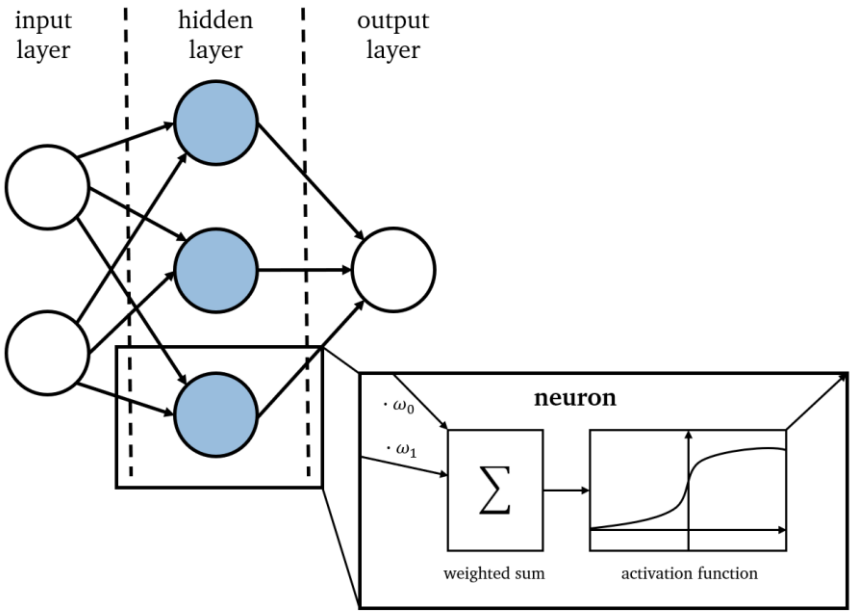


Figure 2.11: ANN with one hidden layer[114]

Each input for a neuron is charged with a weight. The weighted sum of all inputs for a specific neuron goes through an activation function. The output then serves as an input for the next neuron. The choice of activation functions is an important factor in the design of the ANN. Depending on the purpose, different activation functions can be applied, such as the tanh or sigmoid function[115]. Sharma et al.[116] present an overview of activation functions. Those are important as they can present non-linearity to the ANN, enabling it to take on complex, non-linear relationships. The input weights are modified based on how well the prediction matches the training data. That is done through a so-called loss

[112] Cuomo et al. (2022), p. 7.

[113] Aggarwal (2018), p. 106.

[114] Based on Wenzel et al. (2022), p. 3.

[115] Aggarwal (2018), p. 12-13.

[116] Sharma et al. (2020), p. 311-314.

function and is described by Cuomo et al.[117] as the feedback mechanism. The loss function evaluates the accuracy of the ANN based on the training data, often through a linear regression. It compares the predicted value with the actual value. While there are different possibilities to model a loss function, the most common choice across different fields is through a mean square error (MSE)[118]. The goal is to minimize the loss function, meaning to minimize the error. That is done by adjusting the weights, which is also called backpropagation, and poses the basis of the learning or training of the ANN. This optimization algorithm is constantly repeated to find optimal weights to receive accurate predictions.

Using the designations proposed by Cuomo et al. (2022), the PIN represents the functional component of the PINN and overlaps with the NN. It contains the information regarding physical knowledge in form of PDE. PDE depict the non-linear relationship between a variable, such as a parameter, and its partial derivatives to time and space. In the context of PINN, PDE describe the physical knowledge in form of mathematical models. It is important to keep in mind that these mathematical models are only approximations, meaning that PDE are only an incomplete representation of the real complex physical process. Because ANN can derive real relationships from the data sets, the interplay between ANN and PDE can form a more reliable model[119].

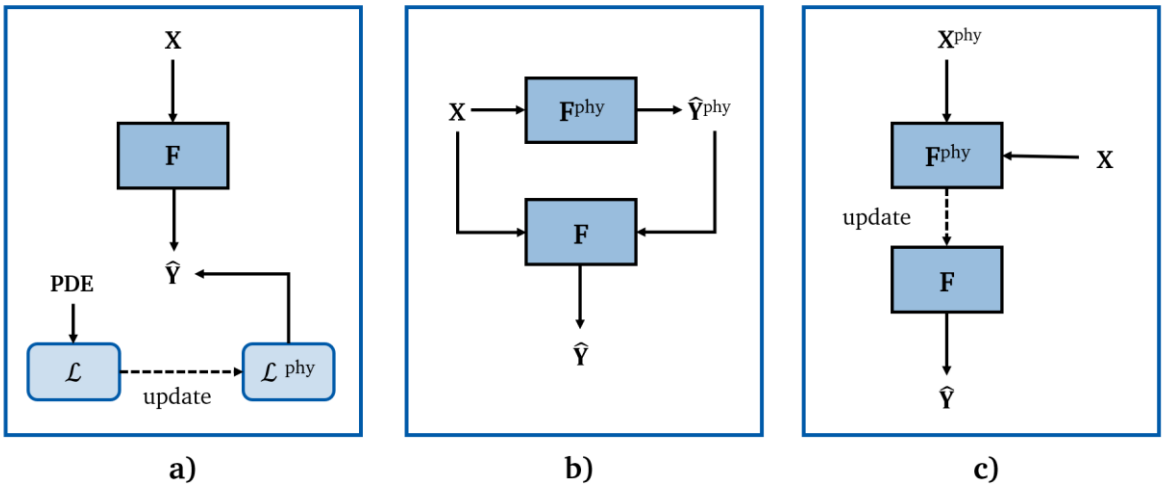


Figure 2.12: Implementation of physics model (a) as constraints in loss function, (b) as additional inputs, (c) for pretraining[120]

In the case of PINN, there are different possibilities to consider PDE. Kapusuzoglu and Mahadevan[121] proposed three different approaches as shown in Figure 2.12. The first one is adding the PDE to the loss function $\mathcal{L}$ as extra terms, updating it to $\mathcal{L}^{phy}$, while the ANN F is trained with the available data

[117] Cuomo et al. (2022), p. 7.

[118] Zhao et al. (2015), p. 2.

[119] Kapusuzoglu and Mahadevan (2020), p. 5.

[120] Based on Kapusuzoglu and Mahadevan (2020), p. 7.

[121] Kapusuzoglu and Mahadevan (2020), p. 5-7.

sets $\mathbf{X}$ obtained through experiments. These physical laws then serve as constraints, so no predictions $\widehat{\mathbf{Y}}$ that violate these constraints are made. The second method uses the outputs of the physics model F^{phy}, meaning the predictions $\widehat{\mathbf{Y}}^{\mathrm{phy}}$, as additional inputs for the ANN F for predictions $\widehat{\mathbf{Y}}$. These outputs are equivalent to the experimental inputs. This method uses the experimental data $\mathbf{X}$ to correct the output $\widehat{\mathbf{Y}}^{\mathrm{phy}}$ of the physics model F^{phy}, as these are only approximations. In the final technique, the physical knowledge is used for the pretraining of the ANN. This means synthetic data $\mathbf{X}^{\mathrm{phy}}$ is generated using the physics models, that data is used to pretrain the ANN F^{phy}, and afterwards the actual training process begins using the experimental data sets $\mathbf{X}$, updating the ANN to F. An advantage of this method is that due to the pretraining, there are no difficulties regarding the initialization of the training. This means the initial weights chosen are already considerably fitted, so the experimental data merely upgrades and fine-tunes the weights, reducing the amount of data needed. Here, the PDE are only used for this initialization and play no further role in the training process. It has been shown that even faulty physics models can reduce the amount of experimental data needed[122]. Another advantage is that the synthetic data from PDE can cover a wide array of data, which might not be possible with expensive experiments. This allows the PINN to have a wider generalization surpassing the range of experimental data[123]. This makes the third method most suitable for the research concerning pure copper LPBF. Of course, combinations of these methods can be made to further improve the effectiveness of the ANN.

To summarize, the goal of the PINN is to predict a solution $\widehat{\mathbf{Y}}$, using the weights $\mathbf{X}$. These weights $\mathbf{X}$ must be chosen, so that $\widehat{\mathbf{Y}}$ is as close to the actual value $\mathbf{Y}$ as possible, thus minimizing the error through the loss function proportional to $(\mathbf{Y} - \widehat{\mathbf{Y}})^2$.

ANN can be differentiated into two different types: feedforward NN and recurrent NN (RNN). The difference between them is that RNN have feedback loops, through which they also consider the information from previous inputs as systematically shown in Figure 2.13. So, instead of a one-dimension information flow from input to output, a RNN uses the output from a previous call again as input for the same node in the NN through these recurrent feedback loops. With every additional input, the so-called memory of the RNN is updated, which allows the network to consider the context and certain behaviors[124]. So every past input is considered in the prediction of RNN.

[122] Jia et al. (2020), p. 4.

[123] Kapusuzoglu and Mahadevan (2020), p. 6.

[124] Salehinejad et al. (2017), p. 1-2.

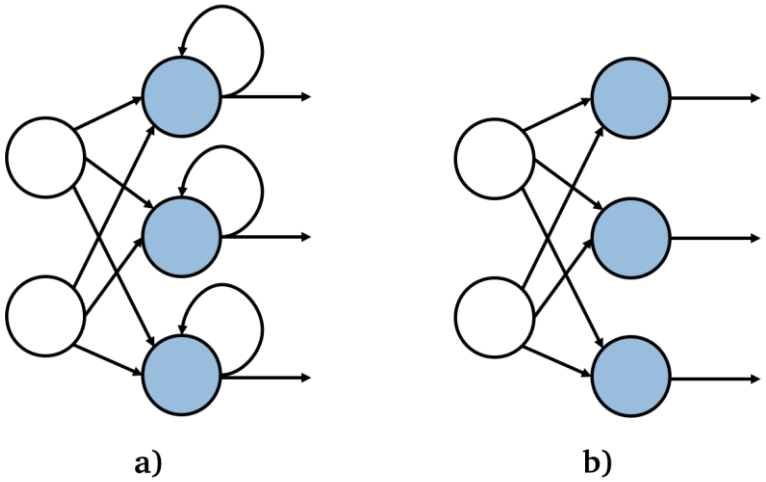


Figure 2.13: a) Recurrent Neural Network, b) Feedforward Neural Network

Wenzel et al. (2022) have proposed a method that aims at optimizing the system reliability in additive manufacturing using PINN. The goal is the ability to predict outcomes and suggest input variables for different requirements. This is so all print defects can be prevented beforehand. This method was tested on Fused Filament Fabrication (FFF) as a case study, and they found that the pretrained PINN had a 50-100 times lower root mean square error (RMSE) when predicting unknown experiments with very few experimental data than the conventional statistical approach. However, while the statistical approach increased in reliability with an increase in experimental data, the PINN approach stopped being effective after a certain amount of training data. They observed that at around 1000 measurements from experiments, the RMSE was the same for both approaches. In the following, a detailed explanation of that method as well as the implication for a different use case is described.

PINN Method

This following method is taken from Wenzel et al.[125] and described in a manner necessary for the application of such on the LPBF process[126].

Firstly, a matter of definition must be clarified. While the original idea of PINN is to use physical models and laws, the same principle applies when using experimental data from other research as well. Through using this previous knowledge for pretraining, less experimental data is needed from oneself, significantly reducing time and money for process development. Wenzel et al.[127] call that prior knowledge Domain Knowledge (DK).

During training, there needs to be a loss function that quantifies how well the estimations match the training data. This loss function calculating the RMSE is here defined as:

[125] Wenzel et al. (2023).

[126] For a more detailed explanation, please refer to Wenzel et al. (2022) and Wenzel et al. (2023).

[127] Wenzel et al. (2022), p. 2.

$$L_{RMSE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \sqrt{\sum \frac{(Y_i - \hat{Y}_i)^2}{n}} \tag{8}$$

Latin Hypercube Sampling (LHS) is a widely used statistical method to generate near-random samples, which are extremely balanced, for experimental design. In this method, LHS samples are used to sample the function space of the data, in order to distinguish them from each other.

Pretraining and Training the PINN

In the first step, the PINN is pretrained using DK. These insights from literature with scientific measurements help the PINN understand which parameters can have an effect. Through interpolation, synthetic data is then generated from that DK and used for pretraining the PINN. This is the third method of implementing prior knowledge, as shown in Figure 2.12(c). The only difference is that the generated data used for pretraining, which usually comes from the physical model, comes from literature experimental data here. However, these experimental data sets can be seen as the physical model here. A more detailed representation of that process is shown in Figure 2.14. $(\mathbf{X} \rightarrow \mathbf{Y})^{\mathrm{phy}}$ describes the physical training data used for pretraining the PINN F^{phy}. Then, using observations $(\mathbf{X} \rightarrow \mathbf{Y})^{\mathrm{obs}}$ through experiments, that pretrained PINN is trained, fine-tuning the PINN. This can be understood as a calibration, updating F^{phy} to the trained PINN F.

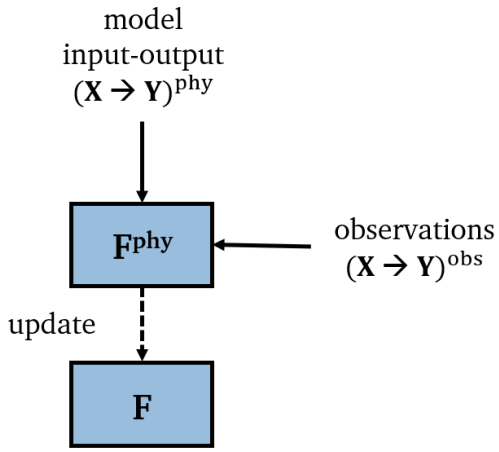


Figure 2.14: Pretraining of the PINN[128]

Data Generation Through SDNN

The following step is data generation from the trained PINN. The idea is that the PINN is retrained for random points $(\mathbf{X} \rightarrow \mathbf{Y})^{\mathrm{ran}}$ to generate new data $\mathbf{Y}^{\mathrm{phy,ran}}$. The retrained PINN model is called F^{ran} and uses randomly generated input $\mathbf{X}^{\mathrm{phy,ran}}$ to generate $\mathbf{Y}^{\mathrm{phy,ran}}$. This is what Wenzel et al. (2023) call Stochastic Data Generation with Neural Networks (SDNN). Through SDNN, huge amounts of data can

[128] Based on Wenzel et al. (2022), p. 5.

be generated that follow the patterns that the original PINN F, and thus the DK and experimental data exhibit. An example can be seen in Figure 2.15, in which three DNN that were trained with the original cubic curve generate data given a random training point. As it is apparent, the generated data follows the pattern of a cubic curve.

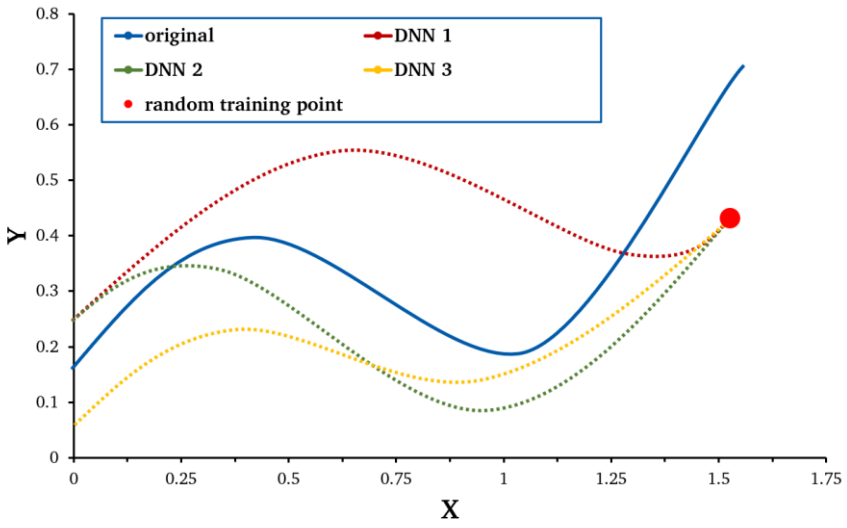


Figure 2.15: Data generation based on random training point[129]

Training the Autoencoder

In the third step, an autoencoder consisting of an encoder and a decoder is trained and used for data compression. Autoencoders are DNN that have a special architecture. They are trained to reconstruct the data $\mathbf{Y}^{\mathrm{phy,ran}}$ as an output $\widehat{\mathbf{Y}}^{\mathrm{phy,ran}}$, which requires much less dimensions. This is done through compression. This means the encoder uses the data $\mathbf{Y}^{\mathrm{phy,ran}}$, which contains many dimensions, to generate a Compressed Feature Vector (CFV), a central layer possessing a small number of dimensions. This CFV represents the behavioral vector $\mathbf{B_a}$.

$$CFV = \boldsymbol{B_a} = E(\boldsymbol{Y}^{phy,ran}) \tag{9}$$

The decoder then uses that behavioral vector to estimate the output data $\widehat{\mathbf{Y}}^{\mathrm{phy,ran}}$.

$$\widehat{\boldsymbol{Y}}^{phy,ran} = D(\boldsymbol{B_a}) \tag{10}$$

Through this, less parameters are needed to receive an estimation. The reason for its use is that the encoder-decoder uses algorithms for supervised learning but applies it on unsupervised learning. This is advantageous because unsupervised learning does not require human labor, with is extensive with big data samples, thus saving time and money.

[129] Based on Wenzel et al. (2023), p. 290.

Training the RNN

A behavioral vector $\widehat{\mathbf{B}}_\mathbf{a}$ is estimated through training a new ANN R. R is a RNN which ensures that all observations $(\mathbf{X} \rightarrow \mathbf{Y})^{\text{obs}}$ of the system directly influence the prediction $\widehat{\mathbf{Y}}$. The RNN is trained using the output data $\widehat{\mathbf{Y}}^{\text{phy,ran}}$ from the autoencoder. The trained RNN then creates an estimation on the general reactions of the system based on the experimental data. These general reactions are called the behavior and defined as:

$$\widehat{\boldsymbol{B}}_{\boldsymbol{a}} = R(\boldsymbol{X}, \boldsymbol{Y}^{\boldsymbol{obs}}) \tag{11}$$

Making the Predictions

In the last step, the actual predictions can be made using the estimated behavioral vector $\widehat{\mathbf{B}}_\mathbf{a}$ and the input parameters $\mathbf{X}$. Using the output data $\widehat{\mathbf{Y}}^{\text{phy,ran}}$, the PINN F is retrained to F^{pred} and makes the predictions $\widehat{\mathbf{Y}}$. Through the optimization algorithm, it is possible to predict the input parameters $\widehat{\mathbf{X}}$ for a desired output $\mathbf{Y}$.

$$\widehat{\mathbf{Y}} = \text{F}^{\text{pred}}(\mathbf{X}, \widehat{\text{Ba}}), \qquad \mathbf{Y} = \text{F}^{\text{pred}}(\widehat{\mathbf{X}}, \widehat{\text{Ba}}) \tag{12}$$

In Figure 2.16 a flow chart summarizing the processes for the prediction of $\widehat{\mathbf{Y}}$ and the relationships between said processes is shown.

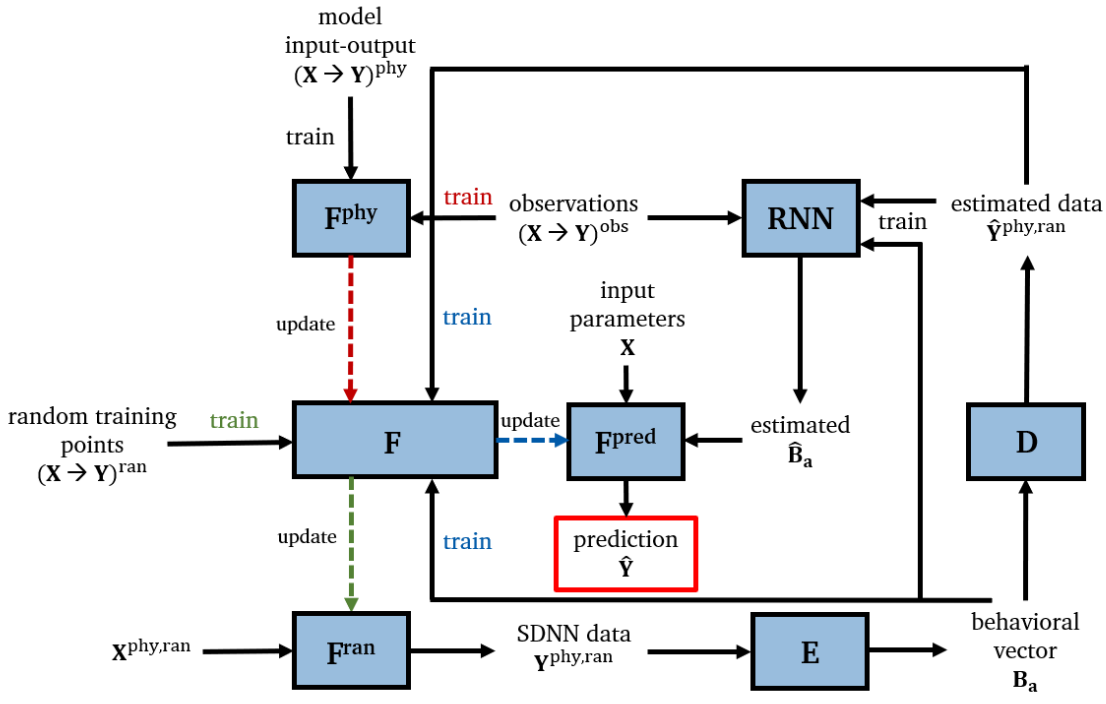


Figure 2.16: Summary of the proposed method

3 Experimental Setup

3.1 Settings of the LPBF Machine

The LPBF process is carried out on the AconityMIDI from the manufacturer Aconity3D, Aconity GmbH. The AconityMIDI is equipped with the Laser System Green Single Mode 200 W, a green laser with a 515 nm wavelength. The technical specifications of the machine are listed in Table 3.1. The laser has a Gaussian intensity distribution and works with a quasi-continuous wave (qcw). Due to an efficiency of $\eta \approx 96.5\ \%$, the maximum effective laser power P_{max} is around 193 W.

Table 3.1: Technical specifications AconityMIDI[130]

Laser System	Green Single Mode 200 W
Laser Power	Max. 200 W
Scanning Speed	Max. 1200 $\frac{\mathrm{mm}}{\mathrm{s}}$
Layer Thickness	Min. 10 µm
Laser Wavelength	515 nm
Laser Spot Diameter	20 – 80 µm
Optics Configuration	F-Theta
Build Space	Ø 170 mm x H 200 mm
Inert Gas Type	Argon 4.6 / 6 bar
Substrate Plate Material	1.4404 (316L) stainless steel, 1.4310 stainless steel

The substrate plate material is stainless steel, which is advantageous, as it leads to a higher absorption in the first layer compared to copper substrates[131]. However, preheating the substrate is not possible, which is why it remains at approximately room temperature.

A Cu-ETP copper powder with 99.95 % purity is used, featuring a particle size distribution (PSD) that ranges between 19 – 33 µm, where the average powder size is 25 µm.

[130] Taken from *Https://aconity3d.Com/products/aconity-Midi/*.

[131] Guan et al. (2019), p. 1389.

3.2 Design of Experiments

Previous experiments using this exact setup with a fixed laser spot diameter of 80 µm, layer thickness of 30 µm, hatch distance of 50 µm and varying values for laser power and scanning speed showed relative density levels as seen in Figure 3.1. The maximum density was achieved using the maximum laser power possible. This process map is not exhaustive, as theoretically, there is still potential to further increase the relative density with e.g. even higher laser power. This process map does not yet give full insight into the optimal parameters for relative density as the maximum has potentially not been exhausted yet. Ideally, the full process map will also show areas of lower relative densities at too-high energy densities due to keyhole porosities. This is why in this study, the laser spot diameter will be decreased to 50 µm, thus reducing the necessary laser power. An exhaustive process map is also essential to achieve samples with different microstructures and thus functional properties, as it allows for a bigger scope to modify process parameters while still maintaining high densities.

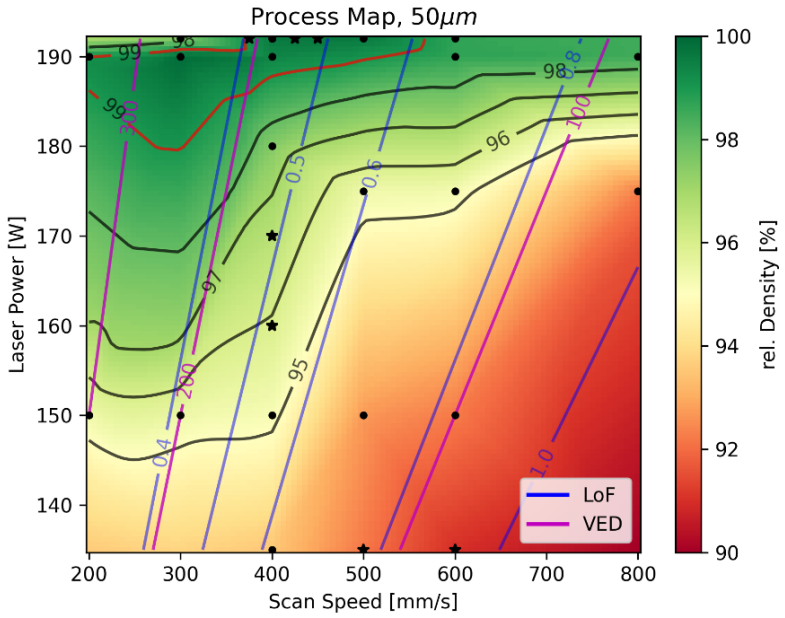


Figure 3.1: Process map with $D = 80$ µm, $t = 30$ µm and $h = 50$ µm

For layer thickness, the smallest thickness that makes sense given the particle size of the copper powder is chosen, which is 30 µm.

A summary of the initial constant process parameters is shown in Table 3.2. This leaves the laser power, scanning speed, and hatch distance as variable parameters that will be modified.

Table 3.2: Constant process parameters

Laser Spot Diameter [µm]	**Layer Thickness** [µm]	**Average Powder Size** [µm]
50	30	25

The goal is not to determine the process parameters to achieve maximum density but rather to determine the process window, in which high densities are maintained while leaving room to adjust process parameters for different microstructures.

A previously conducted study has used both VED and lack of fusion number as an analytical approach to achieve high densities. The insights were limited and both values served as a qualitative parameters at best. While they could be used as a first approach and reference point to limit the experimental room, they were not able to reliably predict the density of the processed sample.

In the present study, an alternative approach is taken. Previous experiments were not able to achieve defects through keyholing and thus could not obtain an exhaustive process map. Furthermore, the lack of fusion number approach based on Tang et al.[132] with Gordon et al.'s[133] melt pool approximations cannot be applied to the keyhole melting mode as it only considers conduction mode geometry and lack of fusion porosities. Jadhav et al.[134] proposed that the keyhole mode must be reached to achieve maximum density. This is why an analytical model is used to calculate the necessary process parameters to reach keyhole mode, the proposed optimal relative density, as well as keyhole porosities.

This could show if using green lasers, high densities could be achieved in both conduction and keyhole mode. If so, the high-density room in which process parameters can be adjusted broadens significantly, opening more opportunities to control the microstructure. It could also give insight into the stability of the keyhole regime when using green lasers. In this case, high stabilities would also broaden opportunities for influencing the microstructure.

The melt pool aspect ratio AR for the keyhole mode as proposed by Fabbro[135] and applied by Jadhav et al.[136] for pure copper can be calculated as follows:

$$AR = \frac{d_m}{w_m} = \frac{R_0}{1 + \frac{v}{v_0}} \tag{13}$$

where

$$R_0 = \frac{A_a * P}{n * d * \kappa * (T_v - T_0)} \tag{14}$$

and

$$v_0 = \frac{2 * n * \kappa}{m * d * \rho * C_p} \tag{15}$$

[132] Tang et al. (2017), p. 3.

[133] Gordon et al. (2020), p. 4-5.

[134] Jadhav et al. (2021), p. 22-23.

[135] Fabbro (2019), p. 347-348.

[136] Jadhav et al. (2021), p. 6.

In the same manner as Jadhav et al.[137], the thermophysical properties of copper at its melting temperature T_m are taken from Mills (2002) as shown in Table 3.3.

Table 3.3: Thermophysical properties of copper at T_m[138]

Property	**Value**	**Unit**
Density ρ	8295	$[\frac{\mathrm{kg}}{\mathrm{m}^3}]$
Heat capacity C_p	469	$[\frac{\mathrm{J}}{\mathrm{kg}*\mathrm{K}}]$
Thermal conductivity κ	330	$[\frac{\mathrm{W}}{\mathrm{m}*\mathrm{K}}]$
Evaporation temperature T_v	2835	[K]
Melting temperature T_m	1357	[K]

Furthermore,

A_a = absorptivity = 0.7, d = laser spot diameter = 50 µm,

T_0 = substrate temperature = 293 K, P = laser power,

v = scanning speed, n_e = 1.5, m_e = 10

In reality, absorptivity is not static and cannot accurately be described with a constant value for different conditions. As previously mentioned, the absorptivity decreases in conduction mode and then increases in keyhole mode. However, as this analytical model aims at investigating the keyhole regime, a value that represents the average absorptivity is chosen. Based on Gruber et al. and Nordet et al.[139], a conservative average absorptivity of 70 % is assumed in this study. For the substrate temperature T_0, a room temperature of 20 °C is assumed, as no preheating takes place.

Here, n_e and m_e are linear coefficients, which depend on the Péclet number P_e [−] based on Fabbro et al.[140]. For pure copper and the calculated P_e of between 0.03 and 0.3 for the given thermophysical properties, LSD of 50 µm, and scanning speed between 100 and 1000 mm/s, the coefficients n and m can be estimated to approximately 1.5 and 10, respectively[141].

[137] Jadhav et al. (2021), p. 6.

[138] Mills (2002), p. 97.

[139] Gruber et al. (2021), p. 6; Nordet et al. (2022), p. 5-6.

[140] Fabbro et al. (2018), p. 3.

[141] Fabbro et al. (2018), p. 6; Jadhav et al. (2021), p. 7.

$$P_e = \frac{v * \rho * C_p * d}{2\kappa} \tag{16}$$

Given these values, the laser power-scanning speed pairs to achieve the optimal aspect ratio of 1.5 can be determined[142]. Furthermore, as an aspect ratio of 1 represents the change from conduction mode to keyhole mode, the laser power-scanning speed pairs to achieve that are calculated as well. This is especially interesting for a LPBF process with green laser, as it gives insight into the achieved density at the transformation between these two modes, showing if the same aspect ratio as for red lasers is needed for maximum density. Table 3.4 shows laser power-scanning speed pairs within the specifications of the AconityMIDI to achieve an aspect ratio of 1 and 1.5. All values marked "-" are scanning speeds that are too low to be seen as relevant.

Table 3.4: Laser power-scanning speed pairs

Aspect Ratio AR = 1		**Aspect Ratio AR = 1.5**	
Laser Power P [W]	**Scanning Speed v [$\frac{mm}{s}$]**	**Laser Power P [W]**	**Scanning Speed v [$\frac{mm}{s}$]**
120	170	120	-
130	225	130	-
140	280	140	-
150	340	150	55
160	395	160	95
170	450	170	130
180	510	180	170
190	565	190	205
193	580	193	220

To ensure a sufficient overlap between the melt tracks, the hatch distance must be close to the LSD or smaller. As mentioned in chapter 2.2, a too small hatch distance will result in porosities as well. This is why three different hatch distances $h = 30;\ 40;\ 50\ \mu m$ are chosen for each sample.

To decide which laser power-scanning speed combinations from Table 3.4 should be printed in the first test series, the following equation is used[143]. It calculates the lowest laser power W_m [W] needed to achieve sufficient melting of the copper powder:

[142] As proposed by Jadhav et al. (2021), p. 10.

[143] Jadhav et al. (2021), p. 18-19; Qu et al. (2021), p. 11.

$$W_m = \frac{\sqrt{\pi} * \kappa * d_e * T_m}{A_{a,c}} \tag{17}$$

with κ = thermal conductivity = 330 $\frac{\mathrm{W}}{\mathrm{m}*\mathrm{K}}$

d_e = laser spot diameter (1/e value) = 35.36 µm

T_m = melting temperature = 1357 K

$A_{a,c}$ = absorptivity (conduction) = 0.5

Then

$$W_m = \frac{\sqrt{\pi} * 330\ \frac{W}{m*k} * 35.36 * 10^{-6}\ m * \ 1357\ K}{0.5} = 56.13\ W$$

It must be considered here that this formula is for a stationary laser and this independent from scanning speed. However, scanning speed has proven to be important. This means with the use of a moving laser such as in LPBF, the minimum laser power required will be higher.

The absorptivity for the conduction mode is taken here to assure that the minimum laser power is calculated for even in the state with the lowest absorptivity. Regardless, the result of this calculation does not provide any insight for this case whatsoever. It is apparent that the influence of the absorptivity is overestimated. This is especially so as the absorptivity is a factor depending on the current state of the melting pool and cannot be determined easily. Therefore, any laser power-scanning speed pairs from Table 3.4 can be taken. The first printed test series is summarized in Table 3.5.

Table 3.5: First test series with $t = 30$ µm, $d = 50$ µm and $D = 25$ µm

#	Laser Power P [W]	Scanning Speed v [$\frac{\mathrm{mm}}{\mathrm{s}}$]	Hatch Distance h [µm]
1.1 – 1.3	120	170	30, 40, 50
1.4 – 1.6	193	580	30, 40, 50
1.7 – 1.9	150	55	30, 40, 50
1.10 – 1.12	193	220	30, 40, 50

The shape of the printed samples is set to 5x5x5 mm cubes. The printing process undergoes a serpentine scanning strategy, where the scanning direction rotates after every layer by 67°. During printing, a support structure is used, so the cubes are not printed directly on the substrate.

3.3 Methodology

Method of Density Measurement

As the goal is to permanently maintain high densities, a reliable method to determine the relative density must be chosen. In literature, many have used the Archimedes method to determine the relative density of the manufactured copper parts, using the theoretical density of 8.93 - 8.96 g/cm^3 of copper as the reference value[144]. Spierings et al.[145] have investigated three different methods to determine the relative density of additive manufactured metals: Archimedes method, micrograph analysis of a cross section and X-ray scanning. When comparing the first two, they found that the Archimedes method is the most reliable. However, the reliability of this method is largely determined on the equipment at hand. Depending on the measuring accuracy of the scale and ambient conditions, the measured relative density can vary substantially. Furthermore, Jadhav et al.[146] have found an overestimation of the measured density in high-porosity samples using the Archimedes method. That is because the liquid-medium will enter the pores in the sample, making the measured volume lower than the real volume.

For this study, micrograph analysis is the most suitable method to determine the relative densities of the samples. For this, the samples are metallographically prepared and the plane parallel to the building direction is examined using the Axiophot microscope with a ZEISS Axiocam 305 color camera. Afterwards, the relative density is determined using the software Fiji ImageJ.

Micrograph analysis can also be used to gain insight into the microstructure of the sample, mainly the size and shape of the grains as well as the pores. This information is crucial for the understanding of the influences on the functional properties.

Metallographic Examination

The procedure to determine the relative density of the samples is done through a metallographic examination. Figure 3.2 shows the steps undertaken.

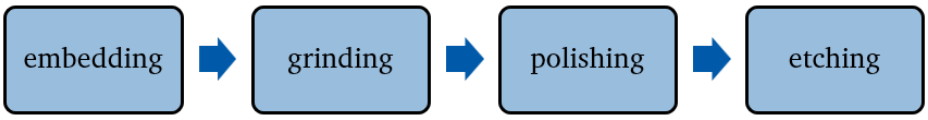


Figure 3.2: Steps for metalligraphic analysis

[144] Bonesso et al. (2021), p. 258; Imai et al. (2020), p. 2; Jadhav et al. (2019), p. 4; Jadhav et al. (2021), p. 4; Yan et al. (2020), p. 6.

[145] Spierings et al. (2011), p. 385.

[146] Jadhav et al. (2021), p. 30.

Before the samples can be prepared metallographically, they are embedded in epoxy resin using the method of cold embedding to ensure protection and better handling. Before this and between every following step, the samples are cleaned in an ultrasonic bath using ethanol.

Table 3.6 shows the grinding steps using sandpaper of different sizes. Between every stage, the angle at which the grinding takes place is rotated by 90°.

Table 3.6: Grinding properties

Grit	**Grain Size** [μm]	**Duration** [s]	**Speed** [rpm]	**Medium**
220	68	180	200	water
320	46	90	200	water
500	30	90	200	water
800	22	90	200	water
1200	15	90	200	water
2400	10	90	200	water

Afterwards, the samples are polished using a diamond paste with the corresponding grain sizes as shown in Table 3.7. The results are completely smooth surfaces that allow the examination of the relative densities.

Table 3.7: Polishing properties

Polishing Plate	**Grain Size** [μm]	**Duration** [s]	**Speed** [rpm]	**Medium**
MOL	6	240	200	aqueous
MOL	3	120	200	aqueous
NAP	1	120	200	aqueous
NAP	0.25	120	200	alcoholic

PINN Methodology

For the application on the LPBF-process with copper, the in chapter 2.4 described method will be divided into two sections and modified accordingly, so that both can function as independent units. This method has previously only been applied to FFF, so this will show the suitability for other cases through minimal modification. The first section consists of the physical model, the pretraining of the PINN, the training of the PINN, and the prediction of the desired data. Here, the trained PINN F and F^{pred} are the same. The second section entails the SDNN, the autoencoder, the RNN, and an updated F^{pred}. Figure 3.3 illustrates the distinction between these sections. This is done because the first section already serves as a functioning unit, capable of making predictions. The second section then includes tools with the goal of further improving these predictions.

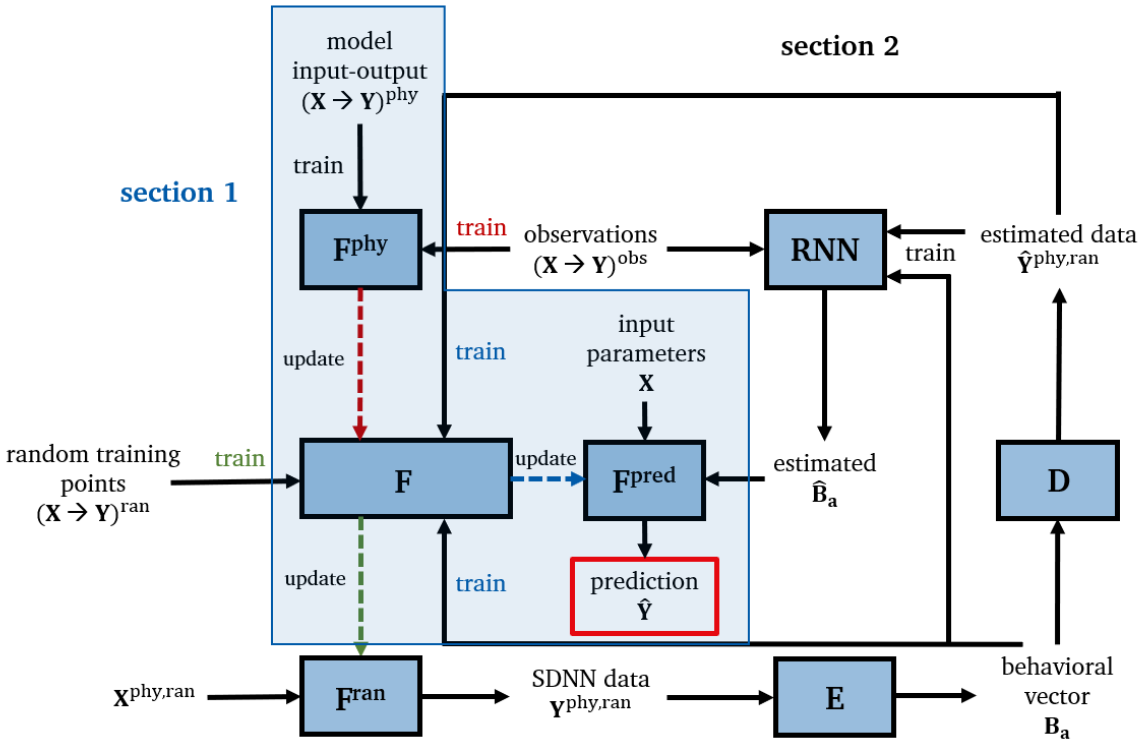


Figure 3.3: Division between sections

Literature has shown that the process with red and green lasers differs substantially, not least because of the difference in absorptivity. Research on green lasers for the LPBF process of pure copper is still extremely limited. Especially when it comes to training data, there is simply not enough experimental data yet to be able to meaningfully train NN in regard to green lasers. While enough experimental data and knowledge has been accumulated for red lasers, it is uncertain whether the transferability of knowledge from red lasers to green lasers is feasible.

That is why the method will be tested and verified first with exclusively red laser data, meaning literature data. An overview of all red laser data used is shown in Table A.3 of Appendix A. For this, the applicability of section 1 is first tested. The available data will be divided into three groups: one for pretraining, one for training, and one to serve as a validation set. In the next step, the transfer of knowledge from the red laser data to green laser data is tested within the scope of section 1. Table A.9

of Appendix provides all green laser data used. Lastly, section 2 will be included to attempt to further improve the predictions of section 1.

In all scenarios, the results of this PINN method will be compared against the traditional and established methods of a simple ANN and Linear Regression. The implementation of all three methods is carried out using Python programming. Appendix B incorporates the relevant Python and Excel files employed in the study. For all three methods, organization charts[147] showing the interplay between the relevant files, as well as the explicit Python code[148] are given to serve as a comprehensive guide to trace the source code and datasets associated with each method[149].

[147] Figure B.1 – Figure B.4 of Appendix B show the organization charts for all methods.

[148] Code B.1 – Code B.8 of Appendix B show the explicit code used.

[149] Table B.1 and Table B.2 of Appendix B provide further information on the implemented files.

4 Results and Discussion

4.1 Machine Performance Insights

Table 4.1 shows the relative densities of the samples in the first test series. What is immediately noticeable is that samples 1.10 and 1.11 could not be analyzed as the given parameter sets did not result in cohesive cubes, but rather lead to significant warping and detachment between layers. Partial warping also occurred in samples 1.4 and 1.12. Examples are depicted in Figure 4.1. Samples 1.5 and 1.6, the remaining two with a laser power of 193 W, possess lower densities than the other samples. At first glance, this might suggest that a laser power of 193 W is too high, implying the use of lower laser powers for the subsequent test series.

Table 4.1: Results test series 1

#	ρ_r [%]	#	ρ_r [%]	#	ρ_r [%]	#	ρ_r [%]
1.1	89.05	1.4	80.38	1.7	90.77	1.10	-
1.2	92.14	1.5	64.75	1.8	95.02	1.11	-
1.3	88.86	1.6	86.27	1.9	91.53	1.12	87.49

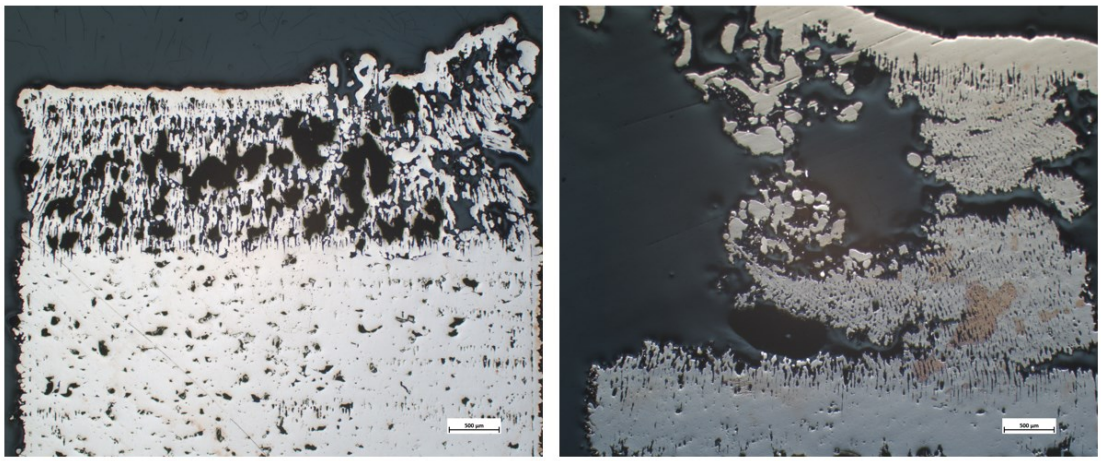


Figure 4.1: Samples 1.4 (left) and 1.11 (right)

To plan the second test series, two approaches are carried out. Firstly, samples 2.1 – 2.8 are parameter sets from Table 3.4, using the laser power-scanning speed pairs with medium-high laser power. Samples 2.1 – 2.5 possess an aspect ratio of 1 and samples 2.6 – 2.8 an aspect ratio of 1.5. The second approach is directly derived from the results of the first test series. As samples 1.2 and 1.8 had the highest densities but still exhibit lack of fusion pores, the same parameter sets with increased laser power are used. These are samples 2.13 – 2.16 and 2.9 – 2.12 respectively. All hatch distances are set

to 40 µm as these have shown to result in the highest relative densities in the previous test series. An overview of all parameter sets and their results can be found in Table A.1 of Appendix A.

The results of the second test series are shown in Table 4.2. Again, two samples were warped to the point of diminishing the ability to measure the density. This concerns sample 2.5 and 2.8, for both of which a laser power of 180 W was used. While this could be again interpreted as a too high laser power, there are further discrepancies.

Table 4.2: Results test series 2

#	ρ_r [%]	#	ρ_r [%]	#	ρ_r [%]	#	ρ_r [%]
2.1	82.86	2.5	-	2.9	91.46	2.13	85.92
2.2	83.97	2.6	94.78	2.10	89.59	2.14	89.03
2.3	94.10	2.7	95.12	2.11	91.15	2.15	87.75
2.4	93.76	2.8	-	2.12	76.33	2.16	86.96

Because samples 2.9 – 2.12 are initially sample 1.8 with gradually higher laser power and sample 1.8 shows lack of fusion pores, it is expected that these samples with have increased densities. The same applies to samples 2.13 – 2.16 in relation to sample 1.2. However, it is apparent that this is not the case. Figure 4.2 shows a side-to-side comparison of sample 1.8 and 2.10, which are identical except for a 10 W increase in sample 2.10. It can be seen that sample 2.10 possesses more lack of fusion porosity than 1.8, even though that is the opposite of what is expected.

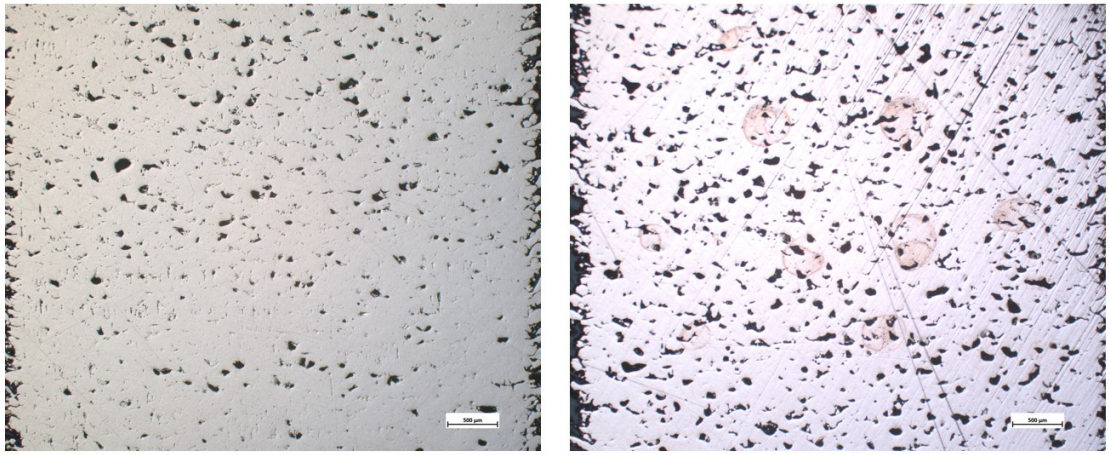


Figure 4.2: Sample 1.8 (left) and 2.10 (right)

Under further investigation of the laser, it comes to light that the set laser power is not actually executed as it shown in Table 4.3. Furthermore, the executed 210.4 W are 17.4 W above the maximum effective laser power $P_{max} = 193$ W of the machine.

Table 4.4 shows the measurements of the laser at set laser powers. It is evident that the executed laser power is far too low on average. Additionally, the maximum laser power is too high for a set laser power of 200 W, this time 23.01 W higher than P_{max}. Further measurements have also shown the lack of response when the laser power is decreased.

While the measured laser powers cannot be quantitatively evaluated due to their dependence on parameters such as the measuring time, these values can still be analyzed qualitatively. All in all, it is safe to say that the machine performance does not meet the necessary standards for a reliable process, which is why the former two test series cannot serve as a representation for the given parameter sets.

The extremely high maximum laser power emitted could also be an explanation for the warping of the samples, while the too low average laser power could explain the remain of lack of fusion pores. Although the defects as seen in Figure 4.2 resemble lack of fusion pores and were previously referred as such, it is important to note that due to the unreliability of the machine, the observed defects cannot be definitely assigned as those.

Table 4.3: Discrepancies to set laser power[150]

Set Laser Power	131.40 W
Minimal Laser Power	0.20 W
Maximal Laser Power	210.40 W

Table 4.4: Measurements of the laser

	Measurement 1	**Measurement 2**	**Measurement 3**	**Measurement 4**
Set Laser Power	50 W	50 W	100 W	200 W
Average Laser Power	37.75 W	36,27 W	71.50 W	125.11 W
Maximal Laser Power	49.80 W	50.32 W	101.44 W	216.01 W

[150] See on display in Figure A.1 of Appendix A.

4.2 PINN Application Using Red Laser

For the first application of the PINN method on the LPBF process, literature data from previous studies with red lasers is used. That literature data is divided into three groups. One group is used as the DK to create a physical model that generates the data for the pretraining of the PINN. The second group is used for the PINN training itself. Normally, the self-conducted experimental test data from the used process would be used here, as the desired application for this method is to predict the outcomes of that specific LPBF process, using only few experimental data. However, in this chapter, the goal is to verify the method, so another group of literature data is used here. The third group will be used as a validation set, to test if predictions from the trained PINN are sensible. For this, again, literature data is used in this study as the outputs of those are already known and can therefore be easily compared with the predicted outputs.

There are six input parameters used in total: laser power, scanning speed, hatch distance, laser spot diameter, layer thickness, and average powder size, which is taken from the powder size distribution. If the value is given in the respective research paper, the d_{50} distribution of the PSD is taken. If not, the mean value of that distribution is taken as an approximation. The goal value or output parameter aimed to be predicted is the relative density.

One variable that needs to be set is the size of the compressed behavioral vector. That can be understood as an informational variable that retains the information about the behavior of the system, which becomes relevant in section 2. However, it must already be defined as it influences the number of neurons for the input layer of the PINN architecture. The aim is to choose a value as low as possible that can still retain all the information. Here, this value is set to 5.

Data Analysis and Regression Model

The following two steps of data analysis and building a suitable regression model are made, before applying section 1 of proposed method on the process of LPBF of pure copper. This is necessary preliminary work, as the data used to train the PINN with is pivotal.

Before using the literature data, a data analysis must be made. This is to ensure that these data are suitable for the training of the PINN. The necessity for this stems from the circumstance that most research studies only publish a small part of their complete test plan, so any outliers might significantly influence the output and therefore skew the results. Furthermore, the distribution of the data can be inspected. The more even and widespread the distribution is, the better it is for the regression model. 160 measurements from 10 different studies are used in total. A complete overview of the research papers and number of measurements taken from each can be found in Table A.2 of Appendix A.

The literature data is split into the previously described three groups using random sampling. From the first group of data, a physical model for the pretraining of the PINN is created. For this, a regression model is built using that group of data, so the data generated from this model follows the patterns of the DK. Apart from the direct effect of the individual input parameters on the relative density, the interaction effects between the input parameters have to be considered as well. Wenzel et al. (2022) call that the main effect and interaction effect, which together make up the total effect. The interaction

effects are extremely important, as they explain the influence of multiple input parameters together on the output.

The reliability of the regression model depends on the number of measurements used to create that model. The available data is extremely limited. In this special case, the data used to create the regression model can be reused for the training of the PINN, as those measurements themselves are not used for the pretraining. Only the data generated from the regression model is used there. 140 measurements are used for both the regression model and the training of the PINN. The remaining 20 measurements will be used for the verification of the method, as illustrated in Figure 4.3[151]. When applying the method on a process in which the experiments are self-conducted, of course only these experimental measurements will be used for the PINN training.

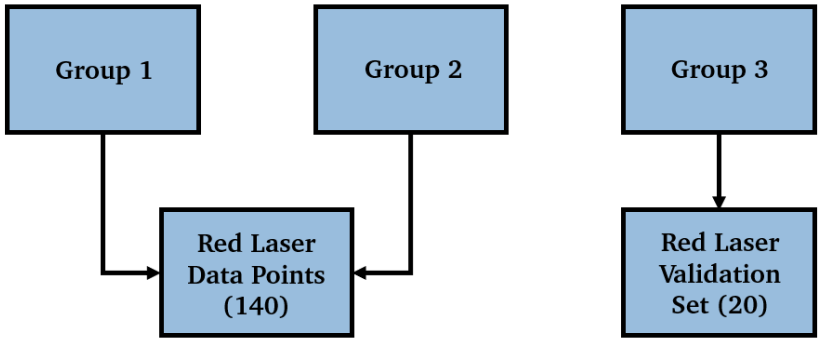


Figure 4.3: Data allocation red laser

The type of regression model used is random forest regression (RFR). The idea behind this method is to use the average of the predictions of multiple decision trees that are generated based on random sampling of the data sets, which is also known as bootstrapping[152]. This ensures that different data is used for every decision tree. These decision trees are independent from each other[153]. Because multiple predictions are used here and then averaged, the RFR produces highly accurate predictions. For the input parameters, random values are generated, which are however restricted within a given range. This range is depicted in Table 4.7.

For the regression model, two main parameters are of importance: the Mean Squared Error (MSE) and R-Squared (R^2), which is the coefficient of determination. The latter describes how much of the variance of the dependent variable (output: relative density) can be explained by the independent variables (input: laser power, scanning speed, hatch distance, laser spot diameter, layer thickness, powder size). The more data is used for the regression model, the better both parameters will be. To determine these parameters, the data is split into two sets: a training set and a testing set. The size of the test set is set to 20 %, so 28 measurements are used for testing. The MSE and R^2 are determined using the

[151] The 20 samples used for validation are depicted in Table A.4 of Appendix A.

[152] *Https://scikit-Learn.Org/stable/modules/generated/sklearn.Ensemble.RandomForestRegressor.Html*.

[153] Breiman (2001), p. 5.

generated outputs through the regression model and the test set. The parameters for this regression model are shown in Table 4.5, which serve as sufficient values for a reliable model.

Table 4.5: Regression parameters

Number of Measurements	Test Split	MSE	R^2
140	0.2	1.78 units	0.91

The summary of the application of the method using the respective data sets is depicted in Figure 4.4.

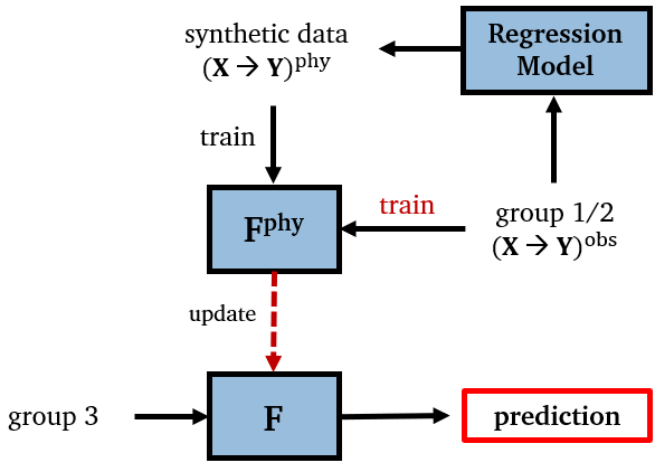


Figure 4.4: Summary section 1 red laser data

Pretraining the PINN

For the pretraining of the PINN, 10000 data points are generated from the regression model and taken as input and output parameters. In the next step, all input and output parameters must be normalized to values [0,1]. When it comes to relative density, the output cannot be linearly normalized as a relative density between 95 % and 99 % makes a huge difference, while relative densities of 70 % and 75 % are equally undesirable. This is why a logarithmic normalization is chosen. For that, the relative density values undergo a logarithmic transformation before being normalized. The percentages are taken as absolute values for the unnormalized data. 70 % relative density is chosen to be the absolute lowest value, as there is no literature data showing densities lower than that. Any values close to 70 % are completely unsatisfactory. 70 % relative density serves as the lower bound and a 100 % relative density as the upper bound. Because values closer to 100 % should be spread more widely than values closer to 70 %, an inversion of the values is carried out. This means initially, values closer to 0 are the higher relative densities and values closer to 1 are the lower relative densities. This is done through subtracting the density values and the upper and lower bound values from 101 within the logarithm function. This is explained by the fact that this results in the transformation of values between [0,1], as $\log(1) = 0$. To reverse the induced inversion, all calculated values are ultimately subtracted from 1.

The normalized output data $\mathbf{Y}^{\text{phy}}$ is calculated as seen in Equation 18. Table 4.6 shows the comparison between unnormalized and normalized data, depicting how relative densities closer to 100 % are more widely spread through the normalization.

$$\mathbf{Y}^{\text{phy}} = \mathbf{Y}^{\text{phy}}_{norm} = 1 - \frac{\log\left(101 - \mathbf{Y}^{\text{phy}}_{unnorm}\right) - \log(101 - 100)}{\log(101 - 70) - \log(101 - 100)} = 1 - \frac{\log\left(101 - \mathbf{Y}^{\text{phy}}_{unnorm}\right)}{\log(31)} \tag{18}$$

Table 4.6: Examples of unnormalized and normalized data

Unnormalized Data	**Normalized Data**
70 %	0
75 %	0.051
95 %	0.478
99 %	0.798
99.9 %	0.972
100 %	1

The input parameters are linearly normalized, each using the highest sensible value as the upper bound *UB* and the lowest sensible value as the lower bound *LB* as shown in Equation 19 and summarized in Table 4.7. Of course, for different applications such as LPBF using a green laser or the LPBF fabrication of other metals, the upper and lower bounds must be adjusted accordingly.

$$\mathbf{X}^{\text{phy}}_{i} = \mathbf{X}^{\text{phy}}_{i,norm} = \frac{\mathbf{X}^{\text{phy}}_{i,unnorm} - LB}{UB - LB} \tag{19}$$

Table 4.7: Lower and upper bounds of input parameters (red laser)

i	**Input Parameter X_i**	**Lower Bound *LB***	**Upper Bound *UB***
1	Laser power	100 W	1200 W
2	Scanning speed	$50\ \frac{\text{mm}}{\text{s}}$	$1200\ \frac{\text{mm}}{\text{s}}$
3	Hatch distance	10 µm	120 µm
4	Laser spot diameter	20 µm	100 µm
5	Layer thickness	10 µm	55 µm

i	Input Parameter X_i	Lower Bound *LB*	Upper Bound *UB*
6	Powder size	15 µm	50 µm

This interpolated and normalized data is used as the synthetic or also called physical training data $(\mathbf{X} \rightarrow \mathbf{Y})^{\mathrm{phy}}$ needed for pretraining the PINN, which results in the PINN F^{phy}. The NN configurations for this step are identical to those used during the training of the PINN and will be described in detail in the corresponding following section.

Training the PINN

For the training or calibration of the PINN, the second set of literature experimental data is used, which in this case are the 140 measurements also used to build the regression model. After normalization, this is the observational data $(\mathbf{X} \rightarrow \mathbf{Y})^{\mathrm{obs}}$.

In the same manner as for the pretraining (see Equation 18 and 19), the input and output variables of the observational data are normalized to values [0,1]. This observational training data $(\mathbf{X} \rightarrow \mathbf{Y})^{\mathrm{obs}}$ is used to train the pretrained PINN F^{phy}, updating it to F.

To train the PINN, there are certain hyperparameters that can be adjusted, as well as the network architecture. The latter describes how the NN is structured, meaning the number of layers, the number of neurons, as well as the activation function in each respective layer. The adjustment of hyperparameters is important as it influences both the time and performance of the training. They are set before the training. For this application, the architecture is set to five layers in total, meaning that there are three hidden layers and each one input and output layer. The reason for this is that the number of layers is dependent on the level of complexity necessary. Restricting the number of layers to a necessary level is important to prevent complications such as an excessive training time or overfitting. A possible approach is to start as simple as possible and gradually increase the number of layers until no further improvements are achieved or even degradation takes place. For all neurons of the hidden and output layer, the sigmoid function is used as the activation function, which is a function existing between 0 and 1 and is very commonly used in NN.

A summary of the PINN architecture can be found in Table 4.8. The number of neurons are chosen through trial and error. The number of neurons in the input layer is determined as the number of features, which is defined as the number of input parameters added with the size of the compressed behavior vector, also called CFV. The output layer is defined to contain one neuron.

However, it must be noted that there are no tangible rules or methods to determine the optimal number of hidden layers and neurons per layer. It always depends on the case and must almost always be determined through a trial-and-error process.

Table 4.8: Architecture of the PINN

Layer	Number of Neurons	Activation Function
Input layer	11	-
Hidden layer 1	32	sigmoid
Hidden layer 2	16	sigmoid
Hidden layer 3	8	sigmoid
Output layer	1	sigmoid

Further selected and adjusted hyperparameters are depicted in Table 4.9. During training, there are two types of loss: training loss and validation loss. The former, also simply known as “loss”, describes how well the estimations match the training data, showing how well the model is learning the patterns of the training data. The latter indicates how well the model generalizes to new, unknown data, using the validation data that was set aside through the validation split. The set validation split value of 20 % is a common share of validation data.

The batch size is the number of training samples used per iteration. This means with a batch size of 32, the training samples are divided by 32, to receive several of batches with each 32 training samples. The chosen batch size of 32 is a relatively small value and thus needs a longer training time, as there are more batches. However, it also results in better generalization and therefore a lower validation loss. This is why it is chosen here. The number of epochs describes how many times the NN algorithm goes through the complete training data set. There, the appropriate number is chosen through observing the point at which the losses stop declining.

Overfitting occurs when the loss is small, but the validation loss is big. This means that the NN performs well on known data but badly regarding unknown data. This is undesirable and must be avoided as the importance of the model lies in the prediction of unknown data. The dropout is a measure to fight overfitting. As overfitting does not pose a significant problem here, the dropout is set to a low value of 5 %.

The learning rate effectively describes how fast the model will converge. It determines how much the model’s parameters should be adjusted in each step. If the learning rate is too low, training time is extremely long, and it might get stuck on a local minimum and not converge at all. However, if the learning rate is too high, the steps might be too big, so the model could converge to the wrong optimum. Through trial and error, the optimal learning rate for the PINN training has proven to be 0.01.

Table 4.9: Hyperparameters for training the PINN

Dropout	Learning Rate	Validation Split	Batch Size	Epochs
0.05	0.01	0.2	32	30

The adjustment of these hyperparameters is called hyperparameter tuning. Even though, there is a general idea of their influences, the fine tuning of these hyperparameters happens through trial-and-error processes depending on the individual application. This applies to the training of all following NN as well.

Prediction and Verification of the Method

The last step of section 1 is the prediction of the model and with it, the verification of the method. While the previous steps served to train the PINN, this step uses the trained PINN and integrates the data wished to be predicted. For this, group 3 of the data is used which consists of 20 data sets. These data sets were not used in the training of the PINN and are therefore completely unknown to the system. This means the process parameters of the validation data sets are used as input for PINN F, which predicts the relative densities of these 20 data points as an output. These predictions are the compared with the actual relative densities from the research studies.

To put the results into perspective, two more methods are used to predict the same set of validation data. First, an ordinary ANN is constructed. This ANN is trained with the 140 data points from group 1/2 to predict the relative densities of the 20 data points from group 3. The same architecture as in Table 4.8 is implemented, with the only difference that the number of epochs is increased to 100, as 30 epochs do not provide sufficient training in this case due to the low amount of training data. As a third method, a simple Linear Regression is done using the 140 data points from group 1/2 to predict the validation set. To ensure that the accuracy of the predictions do not purely rely on chance, the validation set is predicted five times for each method. The mean values of these predictions are then used for the following error measures.

To evaluate the predicted relative densities, the error measure of MSE in percentage points is used as calculated in Equation 20. The results are graphically depicted in Figure 4.5 and the mean MSE and RMSE are shown in Table 4.10. A detailed overview of the individual predictions is depicted in Table A.5 – Table A.8 of Appendix A.

$$MSE = \frac{1}{n} \sum_{t=1}^{n} (y_{pred} - y)^2, \quad RMSE = \sqrt{MSE} \tag{20}$$

Table 4.10: Averaged MSE and RMSE of relative density in percentage points (red laser)

Method	MSE	RMSE	MSE*	RMSE*
PINN	4.58	2.14	3.36	1.83
ANN	14.84	3.85	9.79	3.13

Linear Regression	18.47	4.30	12.95	3.59

*Elimination of number 11 as a potential outlier

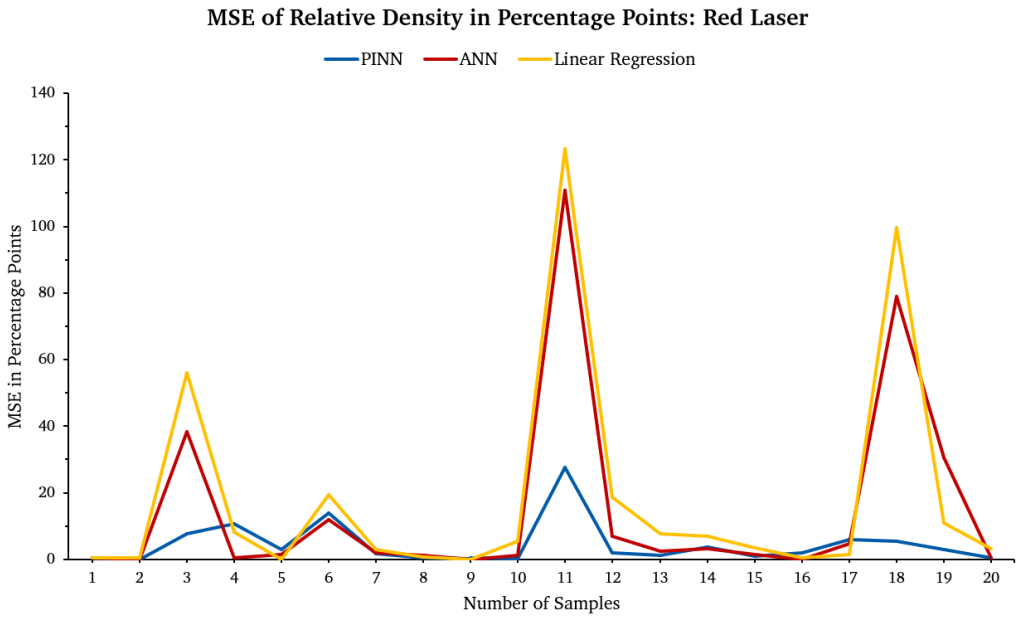


Figure 4.5: Comparison PINN, ANN, Linear Regression (red laser)

It is apparent from Figure 4.5 and the mean error values that the PINN stands out as the most effective method, outperforming the traditional methods ANN and linear regression. This is especially the case for data points that all three methods predict rather poorly. Overall, the four peaks in MSE are consistent between all methods. The PINN method does not exhibit peaks as pronounced as the other two methods. This suggests that the PINN method can handle complexity, as well as potential outliers, better.

Data is extracted from different research publications from varying research environments and system set-ups. The used methodology for measurements differs between research groups. In light of the necessary high precision regarding the analysis of the relative density, the data from the literature can not be fully relied upon.

The values marked with an asterisk in Table 4.10 demonstrate a case in which data point 11 was eliminated to show the difference one potential outlier or measurement error can make. This is done with the purpose of providing an understanding of these measurement uncertainties and to explore the possible thought that the predicted values could be closer to the truth than the literature data.

The RMSE in Table 4.10 can be interpreted as the average deviation, in terms of percentage points, between the estimated relative density and the real relative density of any previously unknown parameter set across the conditions of all ten research papers. Therefore, the PINN RMSE of 2.14 while including all validation data points represents an extremely good outcome given the challenges and complexity involved. Even better results are expected when using self-conducted data for training.

4.3 Application Using Green Laser

The next step is to test the transferability of the method to the printing process with green lasers. It can be tested if the physical knowledge about red lasers can also be used for pretraining the PINN for green lasers. For this, all 160 data points for red lasers will make up group 1, while the data points available for green lasers will be split into group 2 and group 3, as shown in Figure 4.6[154].

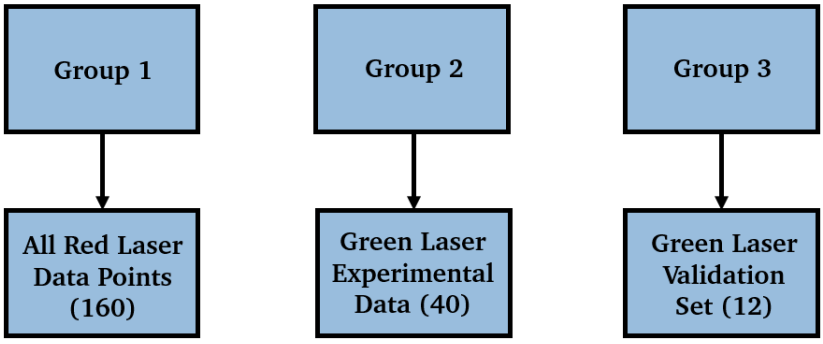


Figure 4.6: Data allocation green laser

The same regression model as in chapter 4.2 is used, this time utilizing all 160 available data points from literature. This results in the regression parameters shown in Table 4.11. The use of these three groups in the PINN process is illustrated in Figure 4.7.

Table 4.11: Regression parameters

Number of Measurements	Test Split	MSE	R^2
140	0.2	1.33 units	0.97

[154] The 12 samples for validation are shown in Table A.10 of Appendix A.

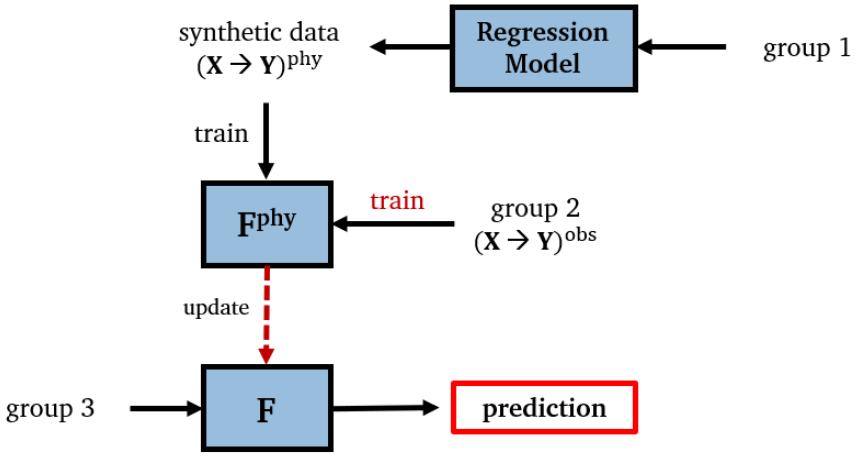


Figure 4.7: Overview application green laser data

The pretraining of the PINN is done in the exact same way as described in the previous chapter, using the 10000 generated and normalized synthetic physical data $(\mathbf{X} \rightarrow \mathbf{Y})^{phy}$ from the upgraded regression model.

For the training of the PINN, group 2 is used which consists of 40 data points taken from an experimental study using a green laser. Due to significant differences in the parameters between studies with red lasers and studies with green lasers, these 40 data points are normalized using the upper and lower bounds shown in Table 4.12. These observational data points $(\mathbf{X} \rightarrow \mathbf{Y})^{obs}$ are then used as input to update F^{phy} to F. Again, both the pretraining and the training of the PINN follow the configurations depicted in Table 4.8 and Table 4.9.

Table 4.12: Lower and upper bounds of input parameters (green laser)

i	Input Parameter X_i	Lower Bound *LB*	Upper Bound *UB*
1	Laser power	100 W	200 W
2	Scanning speed	50 $\frac{mm}{s}$	1200 $\frac{mm}{s}$
3	Hatch distance	30 µm	70 µm
4	Laser spot diameter	40 µm	80 µm
5	Layer thickness	20 µm	40 µm
6	Powder size	20 µm	30 µm

The last step is the prediction of the relative densities of group 3, the validation set. In this case, these are experimental data taken from the same LPBF machine as group 2. This is done in the same manner as for the red lasers, by comparing the predictions of these previously unknown parameter sets to the real relative densities. As before, the traditional methods of ANN and Linear Regression are used as

comparison, with using the mean value of five predictions in each method. The detailed values are depicted in Table A.11 – Table A.14 of Appendix A. The only difference regarding the architecture of the ANN to before is that the number of epochs is further increased to 200, as a lower number does not provide sufficient training given the even lower amount of training data. The results are shown in Table 4.13 and illustrated in Figure 4.8.

Table 4.13: Averaged MSE and RMSE of relative density in percentage points (green laser)

Method	MSE	RMSE
PINN	2.12	1.46
ANN	4.79	2.19
Linear Regression	3.24	1.80

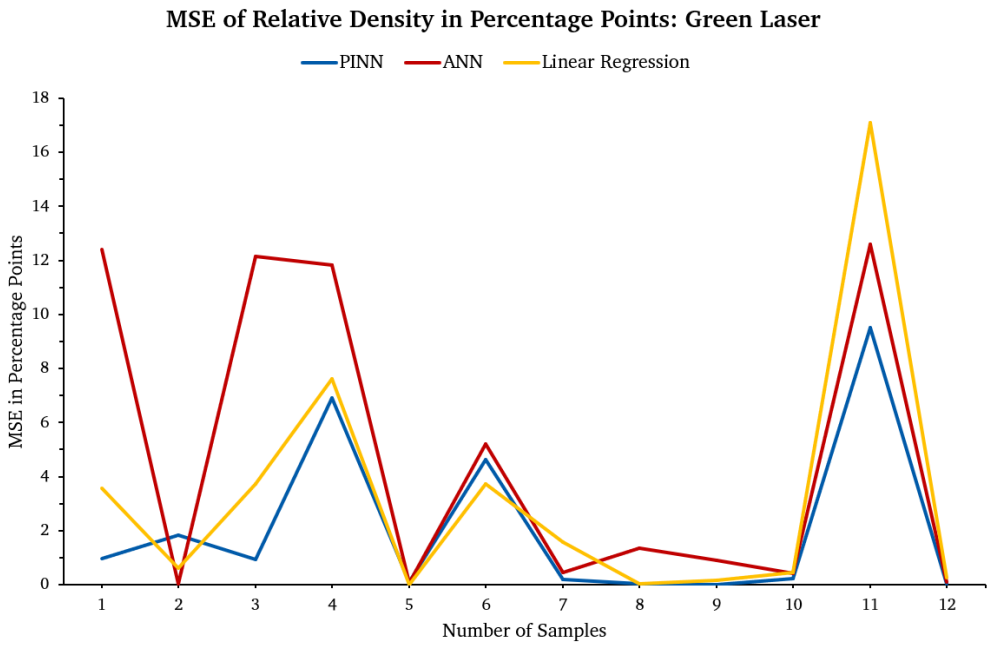


Figure 4.8: Comparison PINN, ANN, Linear Regression (green laser)

The difference in MSE among the three methods is not highly noticeable. A simple linear regression produces excellent predictions that are only slightly inferior compared to those of the PINN method. The reason for this could lie in the lower complexity regarding the experimental data. Examining that data reveals that only three out of the six input parameters exhibit variations, while the other three are constant throughout all data points. This reduces the effective number of input parameters to three. Furthermore, unlike the previous scenario with red laser from literature, all data points in this case stem from the same machine and are subject to the same environmental conditions. This is reflected in the overall lower error values, as evident when comparing the scaling of the ordinate between Figure 4.5 and Figure 4.8.

The PINN method still yields slightly superior predictions with only a mean deviation of 1.46 percentage points, especially when directly compared to the ANN. This shows that the transfer of knowledge from literature data concerning red lasers proves effective in pretraining and ultimately positively influences the training process. This is not only reflected in the need for 170 additional epochs in the simple ANN but also in the accuracy of the predictions.

Reduction of Training Data

As the next step, the amount training data used is reduced to reevaluate the three methods. This is undertaken to observe how these methods react when given less information. It is worth mentioning that this further reduction in training data is only possible because of the previously mentioned low complexity of the available data. If there was data available, in which all input parameters vary, the increased number of parameters and their interaction effects would introduce higher complexity and thus require a much higher number of data points. Figure 4.9 shows the allocation of the used data.

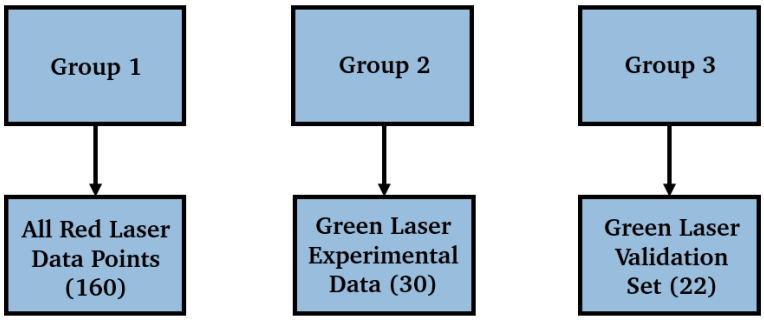


Figure 4.9: Data allocation reduced green laser

The predictions are generated using the same methodology as previously, with unchanged architectures for all three methods. Since only 30 data points are used for training, the remaining 22 are allocated to form a larger validation set[155]. This allows the comparison of more predictions with the actual values.

The results presented in Table 4.14 and Figure 4.10 show a much bigger deviation between the MSE of the PINN method in comparison to the other two methods. Again, the mean of five predictions for all methods is taken. The detailed values of these predictions are shown in Table A.16 – Table A.19 of Appendix A. Similar to the scenario with red lasers, the PINN method significantly outperforms the other methods for data points that have the least accurate predictions throughout all methods.

The error values have increased compared to the previous scenario, which is anticipated. The increase in MSE for the PINN method is marginal, whereas for the ANN and Linear Regression, there is a significant increase. These results highlight a substantial deviation in MSE between the PINN method and

[155] The 22 samples for validation are shown in Table A.15 of Appendix A.

the two traditional methods. While the accuracy of PINN predictions has only slightly decreased to a 1.57 deviation in percentage points, it has considerably decreased for both ANN and Linear Regression.

Regarding the original method proposed by Wenzel et al. (2022), the outcomes show that when provided with a high number of data points, their method yields predictions where the accuracy is comparable to those of a simple ANN. Only for smaller data sets, their method generates predictions that are substantially better than those of a simple ANN. This is consistent with the findings of the current study for section 1 of the PINN.

Table 4.14: Averaged MSE and RMSE of relative density in percentage points (reduced green laser)

Method	MSE	RMSE
PINN	2.46	1.57
ANN	7.29	2.70
Linear Regression	5.19	2.28

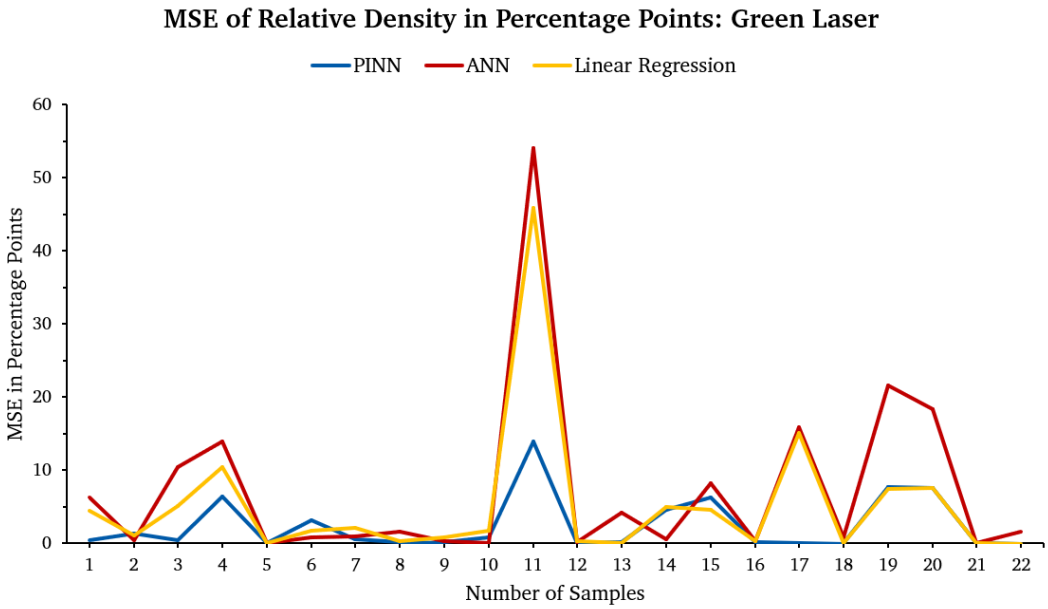


Figure 4.10: Comparison PINN, ANN, Linear Regression (reduced green laser)

In both scenarios involving green laser data, samples 4 and 11 are poorly predicted. In the scenario using reduced green laser data, the predictions for samples 15, 19, and 20 also exhibit higher inaccuracies. Even though the ANN and Linear Regression methods also fall short in predicting other samples, the focus here is on the deficient predictions made by the PINN method, which involve these five data points shown in Table 4.15. The PINN predicted values are taken from the scenario with reduced green laser data.

Table 4.15: Samples with poor predictions (green laser)

i	Laser Power [W]	Scanning Speed $[\frac{mm}{s}]$	Hatch Distance [µm]	Actual Relative Density Y_i [%]	PINN Predicted Relative Density $\hat{Y}_i$ [%]
4	175	800	60	90.80	93.33
11	190	1200	80	85.54	89.28
15	175	600	60	93.59	96.08
19	135	600	50	94.41	91.63
20	135	500	50	95.30	92.55

When comparing the input parameter sets of these data points to similar ones in the training data, it becomes apparent why samples 15, 19, and 20 were predicted by the PINN model as they were. Table 4.16 shows sample 19 compared to data points from the training data that have identical process parameters except for the laser power. An increase in laser power shows an upward trend in relative density. It is therefore expected that sample 19, which has the lowest laser power, will display the lowest density, as it is also predicted by the PINN model. However, the actual experimental measurements contradict this, which results in the deviation between the predicted and actual value. A possible explanation for this could be a measuring error when determining the relative density of the experimental sample. The same logic can be applied to sample 15 and 20, as shown in Table A.20 of Appendix A. This indicates that even with the same LPBF machine and environmental conditions, the results derived from experimental data may not necessarily reflect the truth and must be examined critically.

Table 4.16: Sample 19 in comparison to training data

Laser Power [W]	Scanning Speed $[\frac{mm}{s}]$	Hatch Distance [µm]	Actual Relative Density Y_i [%]
135	**600**	**50**	**94.41**
150	600	50	92.19
175	600	50	95.35
190	600	50	98.66
192	600	50	98.71

The poor prediction for sample 11 could be attributed to the absence of similar parameter sets in the training data. While the trend of the prediction is accurate, the model seems to overestimate the

relative density as it lacks information on even lower values. However, the comparison between models shows that the PINN method was able to deal with this information gap the best.

4.4 Section 2 With Green Laser

In an attempt to further increase the accuracy of the PINN predictions, section 2 is applied analogously to the method by Wenzel et al. as described in chapter 2.4. For this application, the second scenario involving red laser data for pretraining and reduced green laser data for training and predicting is employed. Figure 4.11 shows the complete flow chart involving both section 1 and section 2 using the data allocation from Figure 4.9.

For the retraining of the PINN for data generation, several samples used to sample the function space must be chosen. These samples are chosen through the Latin Hypercube Sampling (LHS) process. Because for this application, a relatively low amount of input parameters is set, the number of LHS samples is set to a lower value as well, namely to 32.

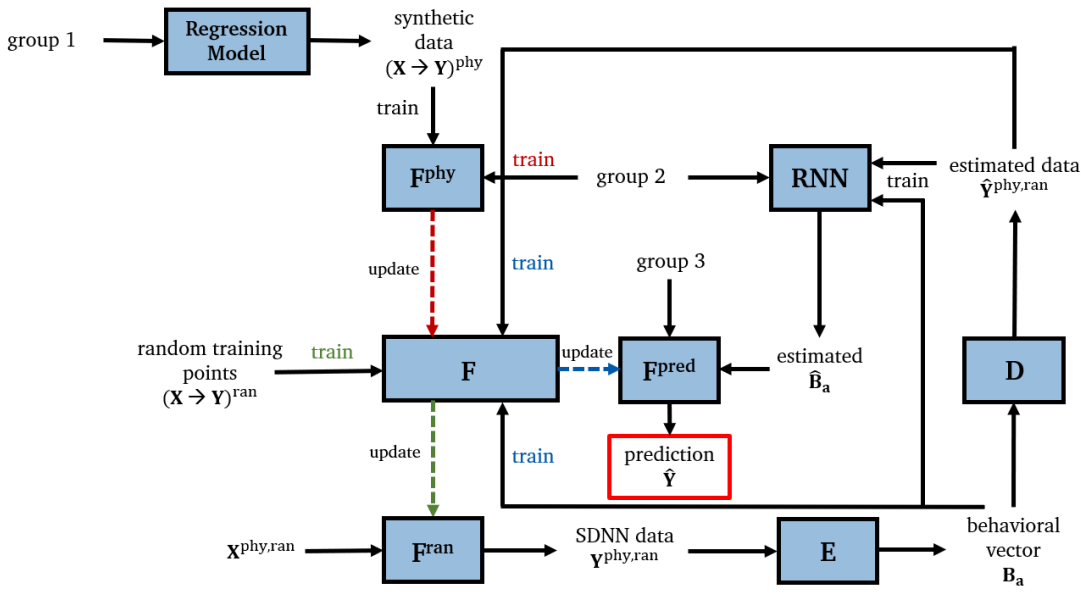


Figure 4.11: Method involving section 2 (reduced green laser data)

Data Generation Through SDNN

Firstly, all steps from section 1 are conducted, however without the last step of predicting the validation set. In the next step, the PINN is retrained, and new data is generated. This model can be called F^{ran}. Several F^{i} are trained for random points, becoming $F^{i,ran}$. The number of random points per training is set to one here. A graphical representation of the training of that model is shown in Figure 2.15. This denotes that the trained models $F^{i,ran}$ follow the patterns of the original PINN F.

It must be understood that not only a single PINN is used in this process. Several PINN F^{i} are trained as described in the previous section, and each F^{i} is then retrained to $F^{i,ran}$ for a random point using the model F^{ran}. The PINN reuse is set to three here, which means that every F^{i} will be retrained three times for three separate random points, leading to $F^{i,ran,1}$, $F^{i,ran,2}$ and $F^{i,ran3}$. Then a new PINN F^{i} is trained, repeating the process.

Using LHS, 32 random sets of weights $\mathrm{X}^{\mathrm{i,phy,ran}}$, representing the input parameters, are generated. The individual models $F^{i,ran}$ produce new data $\mathbf{Y}^{\mathrm{phy,ran}}$ using $\mathbf{X}^{\mathrm{phy,ran}}$ as input. The amount of generated data $\mathbf{Y}^{\mathrm{phy,ran}}$ is set to 10000 here, with 32 $\mathrm{Y}^{\mathrm{i,phy,ran}}$ per $\mathbf{Y}^{\mathrm{phy,ran}}$, corresponding to the LHS produced 32 sets of input parameters $\mathrm{X}^{\mathrm{i,phy,ran}}$. The more samples there are, the higher the reliability when training the following NN.

Table 4.17 show the hyperparameters for the retraining of the PINN for random data points.

Table 4.17: Hyperparameters for training with random data points

Learning Rate	Validation Split	Batch Size	Epochs
0.01	0	1	40

Training the Autoencoder

For the training of the autoencoder, the 10000 samples $\mathbf{Y}^{\mathrm{phy,ran}}$ generated through SDNN are used. The encoder uses all that data to generate the compressed behavioural vector $\mathbf{B_a}$, through which the decoder estimates $\widehat{\mathbf{Y}}^{\mathrm{phy,ran}}$ using much less parameters. The Autoencoder possesses a different architecture with different hyperparameters than the PINN.

This time, the activation function for all neurons in the layers is the tanh function. The number of hidden layers depends on an autoencoder (AE) factor, which determines the factor with which the autoencoder is getting smaller and then bigger from layer to layer. In input and output are both all 32 $\mathrm{Y}^{\mathrm{i,phy,ran}}$ belonging to one $F^{i,ran}$. These are compressed to the size of the compressed behavior vector, which in this application is set to 5. This can be understood as a compression of the behavior of F^{ran}. Here, the AE factor is set to 1.4, which means that there are three layers before reaching the compressed layer and another three layers until the output layer. This is a total of nine layers. The more layers, the higher the complexity and thus the higher the number of parameters. This means that the AE factor must be chosen depending on the amount of SDNN samples $\mathbf{Y}^{\mathrm{phy,ran}}$ at hand. The higher the complexity of the architecture, the more accurate is the output. However, a higher complexity requires more SDNN samples. The output $\widehat{\mathbf{Y}}^{\mathrm{phy,ran}}$ is estimated only using the size of the behavioral vector.

As summarized in Table 4.18, the validation split is again set to the standard value of 20 %. A slightly bigger batch size of 32 is used here, as it yields a slightly lower validation loss. The number of epochs is greatly increased to 200, as the losses continued declining for lower number of epochs. The dropout is set to zero here. The reason for that is with any dropout higher than zero, the validation loss was much lower than the training loss. This can be seen as the opposite of overfitting.

Table 4.18: Hyperparameters for training the autoencoder

Dropout	Validation Split	Batch Size	Epochs	AE Factor
0.0	0.2	32	200	1.4

Training the RNN

In the next step, a RNN is trained using the estimated output data $\hat{\mathbf{Y}}^{\mathrm{phy,ran}}$ and the behavioral vector $\mathbf{B_a}$. The architecture of the RNN also differs from that of the PINN, as the latter is a feed-forward NN. The RNN contains so-called hidden units, namely a Long Short-Term Memory (LSTM) layer with 5 neurons and a Gated Recurrent Unit (GRU) layer with 40 neurons. Both are types of recurrent layers commonly used for RNN.

These hidden layers are implemented to solve the problem of vanishing gradients, which is a recurring problem when it comes to RNN and can even prevent the RNN from further training[156]. Through this, the patterns that have emerged from the observational data from experiments will not be lost during the training of the RNN. Other more rare issues with RNN are exploding gradients, in which the weights are updated in drastic steps. This has severe effects, as it makes the model unstable. This issue can also be solved through the implementation of these hidden units, as they control the information and the degree to which the existing memory should be forgotten[157]. Both LSTM and GRU units possess unique advantages, which is why a combination is used here[158].

The hyperparameters of the RNN are shown in Table 4.19. In the same manner as for the PINN, the dropout is set to 15 % and the validation split to 20 %. The batch size is set to 32 and the RNN trains for 200 epochs.

Table 4.19: Hyperparameters for training the RNN

Dropout	Validation Split	Batch Size	Epochs
0.15	0.2	32	200

Retraining the PINN

In the last step before the predictions for section 2 can be made, the PINN F must be retrained using the behavioral vector $\mathbf{B_a}$ and the through the autoencoder compressed SDNN generated data to make F^{pred}. This updated PINN F^{pred} is then used in the last step to predict the outcomes of the validation set.

[156] Salehinejad et al. (2017), p. 9; Wenzel et al. (2022), p. 4.

[157] Chung et al. (2014), p. 2.

[158] For a detailed comparison regarding similarities and differences please refer to Chung et al. (2014).

The architecture and hyperparameters of the PINN remain the same as in Table 4.8 and Table 4.9.

Prediction and Verification of the Method

For the prediction, not only the updated PINN F^{pred} but also the behavioral vector $\widehat{\mathbf{B}}_{\mathbf{a}}$ of the validation set is needed. The latter can be estimated using the 30 data points from group 2. This behavioral vector $\widehat{\mathbf{B}}_{\mathbf{a}}$ is estimated through the trained RNN.

Finally, using the the behavioral vector $\widehat{\mathbf{B}}_{\mathbf{a}}$ and the inputs of the 22 data points from group 3 as inputs for the PINN F^{pred}, the outputs of the validation set are predicted. Again, these predictions are the compared with the actual relative densities from the research studies. A detailed overview of the predictions is given in Table A.21 of Appendix A. A comparison between the error values of the predictions of section 1 and those with the inclusion of section 2 is displayed in Figure 4.12 and Table 4.20.

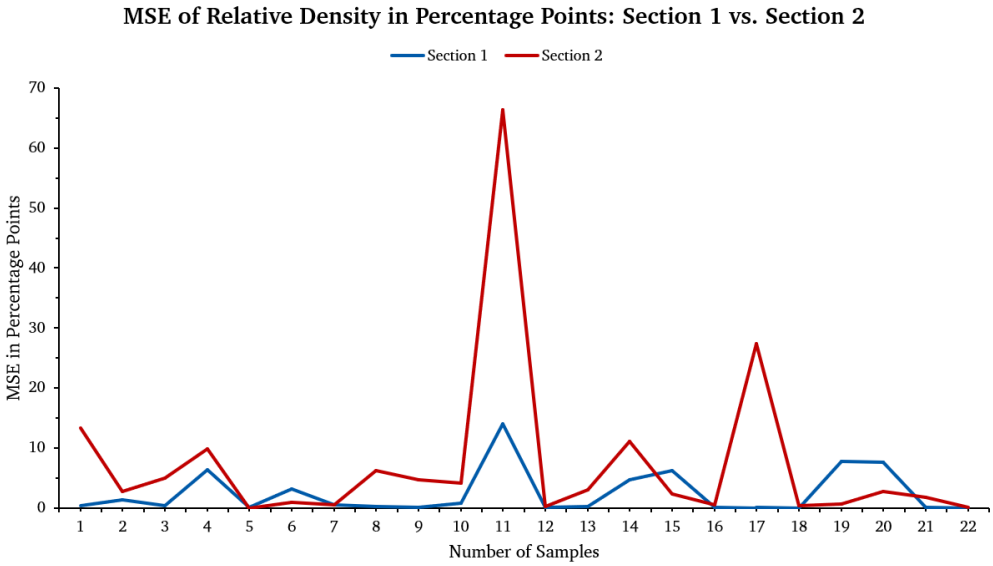


Figure 4.12: Comparison PINN section 1 vs. section 2 (reduced green laser data)

Table 4.20: MSE and RMSE of relative density in percentage points (section 1 vs. section 2)

Method	MSE	RMSE
PINN Section 1	2.46	1.57
PINN Section 2	7.46	2.73

The predictions generated through the additional steps of section 2 have not shown improvement compared those of section 1. In fact, the opposite is true, as there has been a decrease in prediction

accuracy, especially for samples 11 and 17. Figure 4.12 clearly shows that section 1 yields superior results. The MSE of section 2 resembles that of a simple ANN, showing that these complex additional steps are unsuitable for this application.

To understand the possible factors that contribute to the ineffectiveness of section 2, a short comparative analysis between this application and the FFF application from the original method as proposed by Wenzel et al. (2022) is undertaken. The original method corresponds to section 2 of the PINN employed this study.

One potential explanation lies in the difference of complexity between the data used in this study's green laser application and the data used in the FFF application in the original method. The FFF application involves an intricate data set with 75 defined input parameters. Consequently, the training dataset is more extensive, consisting of 1273 measurements[159]. Such as the green laser data, these measurements were conducted using the same FFF machine model and under identical environmental conditions. As established, the effective number of input parameters for green laser data is only three, while using 30 measurements for training. Given the low complexity and low amount of the green laser data, it is likely that overfitting occurs, especially because section 2 of the method entails a high model complexity.

Therefore, it can be concluded that for this application, the proposed method by Wenzel et al. (2023) in its entirety cannot be effectively applied. The results suggest that for the LPBF process with copper, section 2 should be excluded, as the use of the modified section 1 leads to highly satisfactory results.

[159] Wenzel et al. (2022), p. 12.

5 Conclusion and Further Research

The findings of this study establish the superiority of the PINN method over simpler models such as a basic ANN or linear regression in the predictive accuracy of relative densities when given the process parameters. The application of this method was conducted for the LPBF process of copper, utilizing red laser data from literature and experimental green laser data. Three scenarios were tested, in all of which the PINN method yields the most accurate predictions. Notably, the PINN method especially excels in handling data sets that possess high complexity or are very limited. Furthermore, the employment of red laser data for pretraining in scenarios involving green laser data demonstrates that the transfer of knowledge from experiments involving red lasers to experiments with green lasers is feasible.

For the first scenario exclusively involving red laser data from different studies, the PINN method exhibits a deviation of 2.14 percentage points, as opposed to 3.85 and 4.30 percentage points for ANN and linear regression, respectively. Regarding green laser data, the PINN model's predictions show a deviation of 1.46, while ANN and linear regression have deviations of 2.19 and 1.80 percentage points. The disparity increases when using a reduced set of green laser data, as the PINN model's predictive accuracy only experiences a slight decrease to 1.57, while the ANN and linear regression show a more substantial decrease to 2.70 and 2.28 percentage point deviations.

This research addresses a critical gap by pioneering the application of the PINN method, as well as applying other predictive methods to the LPBF process. In doing so, it helps the research in the realm of lean process development.

This thesis further contributes to the field of research, as the PINN method used is a refined version of the PINN method proposed in literature. This applied method is specifically modified to fit the LPBF process of copper, given the available data sets, to predict the relative densities using the process parameters. This method employed in this study outperforms the originally proposed model in its entirety significantly, regarding the accuracy of the predictions, while also minimizing the level of model complexity.

While the LPBF machine used in this study did not work reliably, this study still presents a promising test series that focuses on the geometry of the melt pools and therefore holds meaningful promise for a broader analysis in the context of density, microstructure, and functional properties. The literature research serves as a foundational step that provides the essential knowledge for influencing the microstructure and thus, the mechanical and electrical properties of LPBF copper samples.

In summary, this thesis not only contributes to the current research of the LPBF process with copper, but also aligns with and significantly contributes to the current research gap in the application of machine learning technologies, specifically PINN, on the LPBF process.

While the relatively small data set related to green lasers show the potential and superiority of the PINN method over traditional methods, this study also recognizes the limitations that come with it. Further experiments with higher variability in input parameters will increase the complexity of the data and will likely further enhance the performance of the PINN method compared to other predictive methods. With more comprehensive data sets, the PINN model's capabilities to handle complexity through the interaction between these diverse parameters can be further explored.

This method is not limited to predict only the relative density but can also be used for any other output parameter. This study has provided the first step in helping to adapt the process to complex requirements, such as specific functional properties. The application of this method on mechanical and electrical properties, such as tensile strength and electrical conductivity will greatly reduce the number of experiments needed to achieve the desired outcome. In addition, the effect of input parameter sets on the functional properties, as well as the microstructure of the produced samples can be further understood. For these applications, there are additional possibilities of including physical knowledge to constrict the realm of possible predictions. It is even conceivable to go beyond purely numerical data and to predict the actual picture of the microstructure. Of course, other materials and other processes can also be explored.

For this method to further assist in lean process development, a promising approach would be the direct integration of the model into the LPBF machine itself. This way, certain parameter combinations can be excluded from the beginning through the pretraining and any insights during the printing process can directly contribute to the further training and thus improve the precision of future predictions. This would greatly reduce the trial-and-error process commonly associated with LPBF process development. Furthermore, data from both the machine in use and other machines operating all over the world could be gathered to have all that data at hand, even before conducting any own experiments. This would lead to a continual enhancement of the PINN model at a much faster rate than using only self-conducted the experimental data.

In conclusion, the results and knowledge obtained in the study establish a framework that allow for exciting and promising possibilities in the future for the optimization of the LPBF process. Through the exploration of these suggested research paths, the PINN method will have a meaningful contribution to the future of additive manufacturing.

References

Aggarwal, C. C. (2018). Neural Networks and Deep Learning. *Neural Networks and Deep Learning a Textbook*, 105–167. https://www.academia.edu/42981452/Neural_Networks_and_Deep_Learning_Charu_C_Aggarwal

Bonesso, M., Rebesan, P., Gennari, C., Mancin, S., Dima, R., Pepato, A., & Calliari, I. (2021). Effect of Particle Size Distribution on Laser Powder Bed Fusion Manufacturability of Copper. *BHM Berg- Und Hüttenmännische Monatshefte*, *166*(5), 256–262. https://doi.org/10.1007/s00501-021-01107-0

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1023/A:1010950718922

Calignano, F., Galati, M., & Iuliano, L. (2019). A Metal Powder Bed Fusion Process in Industry: Qualification Considerations. *Machines*, *7*(4), 72. https://doi.org/10.3390/machines7040072

Callister, W. D., & Rethwisch, D. G. (2014). *Materials Science and Engineering: An introduction* (9th ed.). Wiley-Blackwell.

Chowdhury, S., Yadaiah, N., Prakash, C., Ramakrishna, S., Dixit, S., Gupta, L. R., & Buddhi, D. (2022). Laser powder bed fusion: a state-of-the-art review of the technology, materials, properties & defects, and numerical modelling. *Journal of Materials Research and Technology*, *20*, 2109–2172. https://doi.org/10.1016/j.jmrt.2022.07.121

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014, December 11). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. http://arxiv.org/pdf/1412.3555.pdf

Colopi, M., Demir, A. G., Caprio, L., & Previtali, B. (2019). Limits and solutions in processing pure Cu via selective laser melting using a high-power single-mode fiber laser. *The International Journal of Advanced Manufacturing Technology*, *104*(5-8), 2473–2486. https://doi.org/10.1007/s00170-019-04015-3

Cordero, Z. C., Knight, B. E., & Schuh, C. A. (2016). Six decades of the Hall–Petch effect – a survey of grain-size strengthening studies on pure metals. *International Materials Reviews*, *61*(8), 495–512. https://doi.org/10.1080/09506608.2016.1191808

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, *92*(3), 1–62. https://doi.org/10.1007/s10915-022-01939-z

Dai, D., Gu, D., Poprawe, R., & Xia, M. (2017). Influence of additive multilayer feature on thermodynamics, stress and microstructure development during laser 3D printing of aluminum-based material. *Science Bulletin*, *62*(11), 779–787. https://doi.org/10.1016/j.scib.2017.05.007

Fabbro, R. (2019). Scaling laws for the laser welding process in keyhole mode. *Journal of Materials Processing Technology*, *264*, 346–351. https://doi.org/10.1016/j.jmatprotec.2018.09.027

Fabbro, R., Dal, M., Peyre, P., Coste, F., Schneider, M., & Gunenthiram, V. (2018). Analysis and possible estimation of keyhole depths evolution, using laser operating parameters and material properties. *Journal of Laser Applications*, *30*(3), Article 032410. https://doi.org/10.2351/1.5040624

Furukawa, M., Horita, Z., Nemoto, M., Valiev, R. Z., & Langdon, T. G. (1996). Microhardness measurements and the Hall-Petch relationship in an Al□Mg alloy with submicrometer grain size. *Acta Materialia*, *44*(11), 4619–4629. https://doi.org/10.1016/1359-6454(96)00105-X

Gargalis, L., Ye, J., Strantza, M., Rubenchik, A., Murray, J. W., Clare, A. T., Ashcroft, I. A., Hague, R., & Matthews, M. J. (2021). Determining processing behaviour of pure Cu in laser powder bed fusion using direct micro-calorimetry. *Journal of Materials Processing Technology*, *294*, 117130. https://doi.org/10.1016/j.jmatprotec.2021.117130

Gordon, J. V., Narra, S. P., Cunningham, R. W., Liu, H., Chen, H., Suter, R. M., Beuth, J. L., & Rollett, A. D. (2020). Defect structure process maps for laser powder bed fusion additive manufacturing. *Additive Manufacturing*, *36*, 101552. https://doi.org/10.1016/j.addma.2020.101552

Gruber, S., Stepien, L., López, E., Brueckner, F., & Leyens, C. (2021). Physical and Geometrical Properties of Additively Manufactured Pure Copper Samples Using a Green Laser Source. *Materials*, *14*(13), 3642. https://doi.org/10.3390/ma14133642

Gu, D., Shi, Q., Lin, K., & Xi, L. (2018). Microstructure and performance evolution and underlying thermal mechanisms of Ni-based parts fabricated by selective laser melting. *Additive Manufacturing*, *22*, 265–278. https://doi.org/10.1016/j.addma.2018.05.019

Gu, D., Wang, H., Dai, D., Yuan, P., Meiners, W., & Poprawe, R. (2015). Rapid fabrication of Al-based bulk-form nanocomposites with novel reinforcement and enhanced performance by selective laser melting. *Scripta Materialia*, *96*(96), 25–28. https://doi.org/10.1016/j.scriptamat.2014.10.011

Guan, J., Zhang, X., Jiang, Y., & Yan, Y. (2019). Insights into fabrication mechanism of pure copper thin wall components by selective infrared laser melting. *Rapid Prototyping Journal*, *25*(8), 1388–1397. https://doi.org/10.1108/RPJ-06-2018-0143

Hooper, P. A. (2018). Melt pool temperature and cooling rates in laser powder bed fusion. *Additive Manufacturing*, *22*, 548–559. https://doi.org/10.1016/j.addma.2018.05.032

Hosseini, V. A., Shabestari, S. G., & Gholizadeh, R. (2013). Study on the effect of cooling rate on the solidification parameters, microstructure, and mechanical properties of LM13 alloy using cooling curve thermal analysis technique. *Materials & Design*, *50*, 7–14. https://doi.org/10.1016/j.matdes.2013.02.088

https://aconity3d.com/products/aconity-midi/. https://aconity3d.com/products/aconity-midi/

https://de.eos.info/de/3d-druck-material/metalle/kupfer. https://de.eos.info/de/3d-druck-material/metalle/kupfer

https://delva.fi/en/copper/. https://delva.fi/en/copper/

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Hummel, M., Külkens, M., Schöler, C., Schulz, W., & Gillner, A. (2021). In situ X-ray tomography investigations on laser welding of copper with 515 and 1030 nm laser beam sources. *Journal of Manufacturing Processes*, *67*, 170–176. https://doi.org/10.1016/j.jmapro.2021.04.063

Ikeshoji, T.-T., Nakamura, K., Yonehara, M., Imai, K., & Kyogoku, H. (2018). Selective Laser Melting of Pure Copper. *JOM - Journal of the Minerals, Metals and Materials Society*, *70*(3), 396–400. https://doi.org/10.1007/s11837-017-2695-x

Imai, K., Ikeshoji, T.-T., Sugitani, Y., & Kyogoku, H. (2020). Densification of pure copper by selective laser melting process. *Mechanical Engineering Journal*, *7*(2), 19-00272-19-00272. https://doi.org/10.1299/mej.19-00272

Jadhav, S. D., Dadbakhsh, S., Goossens, L., Kruth, J.-P., van Humbeeck, J., & Vanmeensel, K [K.] (2019). Influence of selective laser melting process parameters on texture evolution in pure copper. *Journal of Materials Processing Technology*, *270*, 47–58. https://doi.org/10.1016/j.jmatprotec.2019.02.022

Jadhav, S. D., Goossens, L. R., Kinds, Y., van Hooreweder, B., & Vanmeensel, K [Kim] (2021). Laser-based powder bed fusion additive manufacturing of pure copper. *Additive Manufacturing*, *42*, 101990. https://doi.org/10.1016/j.addma.2021.101990

Jia, X., Willard, J., Karpatne, A., Read, J. S., Zwart, J. A., Steinbach, M., & Kumar, V. (2020, January 28). *Physics-Guided Machine Learning for Scientific Discovery: An Application in Simulating Lake Temperature Profiles*. https://arxiv.org/pdf/2001.11086.pdf

Kapusuzoglu, B., & Mahadevan, S. (2020). Physics-Informed and Hybrid Machine Learning in Additive Manufacturing: Application to Fused Filament Fabrication. *JOM - Journal of the Minerals, Metals and Materials Society*, *72*(12), 4695–4705. https://doi.org/10.1007/s11837-020-04438-4

Karlsson, D., Chou, C.-Y., Pettersson, N. H., Helander, T., Harlin, P., Sahlberg, M., Lindwall, G., Odqvist, J., & Jansson, U. (2020). Additive manufacturing of the ferritic stainless steel SS441. *Additive Manufacturing*, *36*, 101580. https://doi.org/10.1016/j.addma.2020.101580

Li, Y., & Gu, D. (2014). Parametric analysis of thermal behavior during selective laser melting additive manufacturing of aluminum alloy powder. *Materials & Design*, *63*, 856–867. https://doi.org/10.1016/j.matdes.2014.07.006

Lippold, J. C. (2015). *Welding Metallurgy and Weldability*. Wiley. https://onlinelibrary.wiley.com/doi/book/10.1002/9781118960332 https://doi.org/10.1002/9781118960332

Lu, B. (2020). Additive Manufacturing of Copper-based Alloy by Laser Powder Bed Fusion. *Electronic Theses and Dissertations, 2020-*. https://stars.library.ucf.edu/etd2020/94

Lu, L., Shen, Y., Chen, X., Qian, L., & Lu, K. (2004). Ultrahigh strength and high electrical conductivity in copper. *Science (New York, N.Y.)*, *304*(5669), 422–426. https://doi.org/10.1126/science.1092905

Madonna, V., Giangrande, P., & Galea, M. (2018). Electrical Power Generation in Aircraft: Review, Challenges, and Opportunities. *IEEE Transactions on Transportation Electrification*, *4*(3), 646–659. https://doi.org/10.1109/TTE.2018.2834142

Matula, R. A. (1979). Electrical resistivity of copper, gold, palladium, and silver. *Journal of Physical and Chemical Reference Data*, *8*(4), 1147–1298. https://doi.org/10.1063/1.555614

Mills, K. C. (2002). *Recommended Values of Thermophysical Properties for Commercial Alloys* (1. publ). Woodhead Pub.

Moseley, B., Nissen-Meyer, T., & Markham, A. (2020). Deep learning for fast simulation of seismic waves in complex media. *Solid Earth*, *11*(4), 1527–1549. https://doi.org/10.5194/se-11-1527-2020

Nordet, G., Gorny, C., Mayi, Y., Daligault, J., Dal, M., Effernelli, A., Blanchet, E., Coste, F., & Peyre, P. (2022). Absorptivity measurements during laser powder bed fusion of pure copper with a 1 kW cw green laser. *Optics & Laser Technology*, *147*, 107612. https://doi.org/10.1016/j.optlastec.2021.107612

Patel, S., & Vlasea, M. (2020). Melting modes in laser powder bed fusion. *Materialia*, *9*, 100591. https://doi.org/10.1016/j.mtla.2020.100591

Qiu, H., Hanamura, T., & Torizuka, S. (2014). Influence of Grain Size on the Ductile Fracture Toughness of Ferritic Steel. *ISIJ International*, *54*(8), 1958–1964. https://doi.org/10.2355/isijinternational.54.1958

Qu, S., Ding, J., Fu, J., Fu, M., Zhang, B., & Song, X. (2021). High-precision laser powder bed fusion processing of pure copper. *Additive Manufacturing*, *48*, 102417. https://doi.org/10.1016/j.addma.2021.102417

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

Rosenthal, D. (1941). Mathematical Theory of Heat Distribution during Welding and Cutting. *Welding Journal*. https://www.semanticscholar.org/paper/Mathematical-Theory-of-Heat-Distribution-during-and-Rosenthal/589ce78a8ab8d5e52cf50411a428a5b3c6039455

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017, December 29). *Recent Advances in Recurrent Neural Networks*. http://arxiv.org/pdf/1801.01078.pdf

Samy, V. P. N., Schäfle, M., Brasche, F., Krupp, U., & Haase, C. (2023). Understanding the mechanism of columnar–to-equiaxed transition and grain refinement in additively manufactured steel during laser powder bed fusion. *Additive Manufacturing*, *73*, 103702. https://doi.org/10.1016/j.addma.2023.103702

Sharma, S [Siddharth], Sharma, S [Simone], & Athaiya, A. (2020). Activation Functions In Neural Networks. *International Journal of Engineering Applied Sciences and Technology*, 310–316. https://www.semanticscholar.org/paper/ACTIVATION-FUNCTIONS-IN-NEURAL-NETWORKS-Sharma-Sharma/50fdddde4a86fc6f2e23fd7ccbf60e6a37fc4c6d

Silbernagel, C., Gargalis, L., Ashcroft, I., Hague, R., Galea, M., & Dickens, P. (2019). Electrical resistivity of pure copper processed by medium-powered laser powder bed fusion additive manufacturing for use in electromagnetic applications. *Additive Manufacturing*, *29*, 100831. https://doi.org/10.1016/j.addma.2019.100831

Spierings, A. B., Schneider, M., & Eggenberger, R. (2011). Comparison of density measurement techniques for additive manufactured metallic parts. *Rapid Prototyping Journal*, *17*(5), 380–386. https://doi.org/10.1108/13552541111156504

Tang, M., Pistorius, P. C., & Beuth, J. L. (2017). Prediction of lack-of-fusion porosity for powder bed fusion. *Additive Manufacturing*, *14*, 39–48. https://doi.org/10.1016/j.addma.2016.12.001

Tertuliano, O. A., DePond, P. J., Doan, D., Matthews, M. J., Gu, X. W., Cai, W., & Lew, A. J. (2022). Nanoparticle-enhanced absorptivity of copper during laser powder bed fusion. *Additive Manufacturing*, *51*, 102562. https://doi.org/10.1016/j.addma.2021.102562

Trevisan, F., Calignano, F., Lorusso, M., Lombardi, M., & Fino, P. (2017). Selective laser melting of chemical pure copper. In *Euro PM 2017*. https://www.researchgate.net/publication/320372540_Selective_laser_melting_of_chemical_pure_copper

Wenzel, S., Slomski-Vetter, E., & Melz, T. (2022). Optimizing System Reliability in Additive Manufacturing Using Physics-Informed Machine Learning. *Machines*, *10*(7), 525. https://doi.org/10.3390/machines10070525

Wenzel, S., Slomski-Vetter, E., & Melz, T. (2023). *Zuverlässigkeitsoptimierung in der additiven Fertigung durch physical informed neural networks (PINN)* (No. 2409). *VDI-Berichte*, pp. 283–297.

Yan, X., Chang, C., Dong, D., Gao, S., Ma, W., Liu, M., Liao, H., & Yin, S. (2020). Microstructure and mechanical properties of pure copper manufactured by selective laser melting. *Materials Science and Engineering: A*, *789*, 139615. https://doi.org/10.1016/j.msea.2020.139615

Yap, C. Y., Chua, C. K., Dong, Z. L., Liu, Z. H., Zhang, D. Q., Loh, L. E., & Sing, S. L. (2015). Review of selective laser melting: Materials and applications. *Applied Physics Reviews*, *2*(4), Article 041101. https://doi.org/10.1063/1.4935926

Zhao, H., Gallo, O., Frosio, I., & Kautz, J. (2015, November 28). *Loss Functions for Neural Networks for Image Processing*. http://arxiv.org/pdf/1511.08861.pdf

Appendix A

Table A.1: Overview test series with $t = 30$ µm, $d = 50$ µm and $D = 25$ µm

#	Laser Power P [W]	Scanning Speed v [$\frac{mm}{s}$]	Hatch Distance h [µm]	Relative Density ρ_r [%]
1.1	120	170	30	89.05
1.2	120	170	40	92.14
1.3	120	170	50	88.86
1.4	193	580	30	80.38
1.5	193	580	40	64.75
1.6	193	580	50	86.27
1.7	150	55	30	90.77
1.8	150	55	40	95.02
1.9	150	55	50	91.53
1.10	193	220	30	-
1.11	193	220	40	-
1.12	193	220	50	87.49
2.1	140	280	40	82.86
2.2	150	340	40	83.97
2.3	160	396	40	94.1
2.4	170	450	40	93.76
2.5	180	510	40	-
2.6	160	95	40	94.78
2.7	170	130	40	95.12
2.8	180	170	40	-
2.9	155	55	40	91.46
2.10	160	55	40	89.59
2.11	165	55	40	91.15
2.12	170	55	40	76.33

#	Laser Power P [W]	Scanning Speed v [$\frac{mm}{s}$]	Hatch Distance h [µm]	Relative Density ρ_r [%]
2.13	125	170	40	85.92
2.14	130	170	40	89.03
2.15	135	170	40	87.75
2.16	140	170	40	86.96

Figure A.1: Laser display showing discrepancies between set and executed laser power

Table A.2: Overview data from literature with numbering from Table A.3

Research Paper	Number of Measurements
Effect of Particle Size Distribution on Laser Powder Bed Fusion Manufacturability of Copper (2021) by Bonesso et al. (2021)	48 (1 – 48)
Laser-based powder bed fusion additive manufacturing of pure copper" (2021) by Jadhav et al. (2021)	24 (49 – 72)
Additive Manufacturing of Copper-based Alloy by Laser Powder Bed Fusion (2020) by B. Lu (2020)	17 (73 – 89)
Microstructure and mechanical properties of pure copper manufactured (2020) by Yan et al. (2020)	17 (90 – 106)
Densification of pure copper by selective laser melting process (2020) by Imai et al. (2020)	15 (107 – 121)

Research Paper	Number of Measurements
High-precision laser powder bed fusion processing of pure copper (2021) by Qu et al. (2021)	14 (122 – 135)
Insights into fabrication mechanism of pure copper thin wall components by selective infrared laser melting (2019) by Guan et al. (2019)	9 (136 – 144)
Selective Laser Melting of Pure Copper (2018) by Ikeshoji et al. (2018)	6 (145 – 150)
Nanoparticle-enhanced absorptivity of copper during laser powder bed fusion (2022) by Tertuliano et al. (2022)	6 (150 – 156)
Influence of selective laser melting process parameters on texture evolution in pure copper (2019) by Jadhav et al. (2019)	4 (157 – 160)

Table A.3: Red laser data from literature

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
1	370	400	110	82	20	19.5	97.40
2	370	200	80	82	20	19.5	96.80
3	370	200	90	82	20	19.5	96.50
4	370	200	100	82	20	19.5	96.30
5	370	500	80	82	20	19.5	96.80
6	370	300	110	82	20	19.5	97.00
7	370	500	90	82	20	19.5	97.20
8	370	500	100	82	20	19.5	98.00
9	370	500	110	82	20	19.5	97.20
10	370	200	80	82	20	27.4	92.10
11	370	200	90	82	20	27.4	92.50
12	370	200	100	82	20	27.4	93.20
13	370	300	80	82	20	27.4	91.50
14	370	300	90	82	20	27.4	92.20
15	370	300	100	82	20	27.4	93.40

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
16	370	400	110	82	20	27.4	92.50
17	370	500	80	82	20	27.4	92.50
18	370	500	90	82	20	27.4	92.40
19	370	500	100	82	20	27.4	91.90
20	370	500	110	82	20	27.4	92.00
21	370	200	80	82	20	18.7	93.40
22	370	200	90	82	20	18.7	93.30
23	370	200	10	82	20	18.7	92.70
24	370	200	110	82	20	18.7	94.10
25	370	300	110	82	20	18.7	93.90
26	370	400	80	82	20	18.7	93.60
27	370	400	90	82	20	18.7	94.00
28	370	400	100	82	20	18.7	94.00
29	370	400	110	82	20	18.7	93.20
30	370	500	80	82	20	18.7	93.70
31	370	500	90	82	20	18.7	93.20
32	370	500	100	82	20	18.7	96.60
33	370	500	110	82	20	18.7	97.60
34	370	200	110	82	20	19.5	96.40
35	370	300	80	82	20	19.5	97.60
36	370	300	90	82	20	19.5	96.40
37	370	300	100	82	20	19.5	97.40
38	370	400	80	82	20	19.5	97.20
39	370	300	110	82	20	27.4	92.50
40	370	400	80	82	20	27.4	92.50
41	370	400	90	82	20	27.4	92.60

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
42	370	400	100	82	20	27.4	92.40
43	370	300	80	82	20	18.7	93.20
44	370	300	90	82	20	18.7	94.10
45	370	300	100	82	20	18.7	93.40
46	370	200	110	82	20	27.4	91.90
47	370	400	90	82	20	19.5	98.30
48	370	400	100	82	20	19.5	98.10
49	200	100	90	37.5	30	44.4	89.10
50	200	200	90	37.5	30	44.4	88.50
51	200	400	90	37.5	30	44.4	89.70
52	200	600	90	37.5	30	44.4	89.00
53	200	800	90	37.5	30	44.4	87.80
54	200	1000	90	37.5	30	44.4	90.60
55	300	100	90	37.5	30	44.4	97.10
56	300	200	90	37.5	30	44.4	97.50
57	300	400	90	37.5	30	44.4	96.30
58	300	600	90	37.5	30	44.4	94.50
59	300	800	90	37.5	30	44.4	91.90
60	300	1000	90	37.5	30	44.4	89.00
61	400	100	90	37.5	30	44.4	94.30
62	400	200	90	37.5	30	44.4	96.10
63	400	400	90	37.5	30	44.4	98.10
64	400	600	90	37.5	30	44.4	98.70
65	400	800	90	37.5	30	44.4	98.20
66	400	1000	90	37.5	30	44.4	92.90
67	500	100	90	37.5	30	44.4	95.50

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
68	500	200	90	37.5	30	44.4	96.00
69	500	400	90	37.5	30	44.4	98.10
70	500	600	90	37.5	30	44.4	99.00
71	500	800	90	37.5	30	44.4	99.30
72	500	1000	90	37.5	30	44.4	92.60
73	350	50	120	100	30	42.65	86.50
74	350	100	120	100	30	42.65	87.00
75	350	150	120	100	30	42.65	87.00
76	350	200	120	100	30	42.65	87.70
77	350	300	120	100	30	42.65	86.20
78	350	500	120	100	30	42.65	87.50
79	350	700	120	100	30	42.65	86.00
80	350	800	120	100	30	42.65	86.50
81	300	50	120	100	30	42.65	81.50
82	300	100	120	100	30	42.65	84.50
83	300	150	120	100	30	42.65	85.20
84	300	200	120	100	30	42.65	84.80
85	300	300	120	100	30	42.65	84.10
86	300	500	120	100	30	42.65	83.20
87	200	50	120	100	30	42.65	79.90
88	200	100	120	100	30	42.65	79.70
89	200	200	120	100	30	42.65	81.50
90	150	1000	80	100	30	31.2	82.00
91	150	800	80	100	30	31.2	84.00
92	150	600	80	100	30	31.2	88.00
93	150	400	80	100	30	31.2	96.50

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
94	150	200	80	100	30	31.2	96.00
95	200	1000	80	100	30	31.2	86.00
96	200	600	80	100	30	31.2	93.00
97	200	400	80	100	30	31.2	98.50
98	200	200	80	100	30	31.2	92.00
99	250	800	80	100	30	31.2	92.00
100	250	600	80	100	30	31.2	97.00
101	250	400	80	100	30	31.2	97.00
102	300	1000	80	100	30	31.2	91.00
103	350	1000	80	100	30	31.2	95.50
104	350	800	80	100	30	31.2	98.00
105	350	600	80	100	30	31.2	98.00
106	350	400	80	100	30	31.2	95.00
107	600	300	50	100	50	28	98.50
108	700	300	50	100	50	28	99.50
109	800	300	25	100	50	28	95.70
110	800	300	40	100	50	28	95.90
111	800	300	75	100	50	28	96.40
112	800	300	100	100	50	28	96.60
113	800	300	120	100	50	28	96.00
114	800	600	50	100	50	28	98.60
115	800	900	50	100	50	28	98.50
116	800	1200	50	100	50	28	98.50
117	900	300	50	100	50	28	99.50
118	900	600	50	100	50	28	98.50
119	1000	600	50	100	50	28	99.50

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
120	1000	900	50	100	50	28	99.50
121	1000	1200	50	100	50	28	98.50
122	140	600	50	25	10	17.9	92.00
123	160	600	50	25	10	17.9	95.30
124	180	600	50	25	10	17.9	96.90
125	200	600	50	25	10	17.9	99.60
126	220	600	50	25	10	17.9	99.75
127	240	600	50	25	10	17.9	99.65
128	200	200	50	25	10	17.9	98.90
129	200	400	50	25	10	17.9	99.25
130	200	600	50	25	10	17.9	99.60
131	200	800	50	25	10	17.9	97.60
132	200	600	10	25	10	17.9	97.60
133	200	600	30	25	10	17.9	98.75
134	200	600	50	25	10	17.9	99.60
135	200	600	70	25	10	17.9	98.60
136	190	400	60	80	20	34	82.00
137	190	500	60	80	20	34	79.00
138	190	600	60	80	20	34	79.00
139	190	700	60	80	20	34	78.00
140	170	400	60	80	20	34	78.00
141	170	500	60	80	20	34	76.00
142	170	600	60	80	20	34	77.00
143	150	400	60	80	20	34	79.00
144	150	500	60	80	20	34	79.00
145	800	300	25	100	50	28.2	95.70

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
146	800	300	40	100	50	28.2	95.90
147	800	300	50	100	50	28.2	95.50
148	800	300	75	100	50	28.2	96.40
149	800	300	100	100	50	28.2	96.60
150	800	300	120	100	50	28.2	96.00
151	100	300	80	60	50	27.5	85.00
152	400	300	80	60	50	27.5	93.50
153	200	600	80	60	50	27.5	89.00
154	300	600	80	60	50	27.5	92.00
155	400	600	80	60	50	27.5	96.75
156	500	600	80	60	50	27.5	98.00
157	300	400	50	40	30	45	84.00
158	800	400	90	40	30	45	98.00
159	800	400	70	40	30	45	98.00
160	600	200	90	40	30	45	98.00

Table A.4: Random samples from Table A.3 for red laser validation set

#				
5	15	21	38	57
60	73	78	85	88
89	93	101	102	111
117	124	151	155	160

Table A.5: PINN predictions (red laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	86.59	85.50	86.01	86.91	88.80
2	95.62	97.40	96.89	97.01	96.93
3	88.06	88.49	85.78	90.34	86.29
4	94.50	93.97	94.38	93.89	94.63
5	98.78	99.07	98.95	98.62	97.85
6	82.67	82.73	82.77	84.38	84.60
7	85.10	84.68	84.50	85.60	87.03
8	99.11	98.68	98.32	98.99	98.60
9	84.64	85.77	85.40	85.70	87.31
10	95.41	96.51	96.28	96.52	96.69
11	90.70	91.04	90.47	92.85	91.12
12	94.65	95.20	95.60	94.79	94.46
13	95.33	96.29	96.08	96.40	96.49
14	82.89	82.57	82.60	84.44	84.72
15	96.43	98.10	97.04	97.52	97.96
16	94.93	94.39	95.03	94.85	94.85
17	95.17	95.94	95.76	96.25	96.16
18	94.95	93.95	95.04	94.78	94.47
19	90.87	91.41	88.60	91.67	91.04
20	97.12	97.51	97.97	97.09	96.90

Table A.6: ANN predictions (red laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	87.64	87.71	86.85	86.25	85.98
2	96.59	96.84	96.61	96.42	96.42

#	Output 1	Output 2	Output 3	Output 4	Output 5
3	91.24	91.05	91.60	91.00	91.08
4	90.66	90.59	90.38	89.78	89.89
5	98.25	98.22	98.12	98.06	97.98
6	84.41	85.06	82.57	82.74	81.04
7	86.40	86.64	85.30	84.90	84.26
8	98.51	98.53	98.41	98.27	98.35
9	86.96	87.95	86.02	85.50	85.21
10	95.92	96.00	95.75	95.50	95.58
11	86.74	86.90	85.84	85.50	84.89
12	93.90	93.88	93.72	93.30	93.59
13	95.80	95.86	95.64	95.38	95.48
14	84.53	85.16	82.75	82.88	81.22
15	97.72	97.90	97.62	97.43	97.55
16	93.94	93.81	93.74	93.28	93.52
17	95.71	95.74	95.58	95.29	95.45
18	88.66	88.56	88.19	87.58	87.59
19	94.76	94.91	94.51	94.23	94.27
20	97.46	97.62	97.20	97.02	97.12

Table A.7: Linear Regression predictions (red laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	86.64	86.60	87.32	86.77	86.40
2	97.48	97.49	97.44	97.82	97.29
3	92.71	92.61	92.18	92.62	92.28
4	88.12	87.88	88.81	88.24	87.63
5	96.86	97.42	97.38	97.13	97.23

#	Output 1	Output 2	Output 3	Output 4	Output 5
6	84.16	84.13	84.37	83.96	83.89
7	85.79	85.76	86.25	85.78	85.56
8	98.69	98.69	98.55	98.76	98.84
9	86.48	86.54	86.75	86.39	86.40
10	94.38	94.49	94.60	94.56	94.40
11	85.58	85.39	85.61	85.26	85.12
12	91.63	92.11	92.43	91.90	91.90
13	94.35	94.48	94.47	94.47	94.40
14	84.19	84.14	84.50	84.05	83.89
15	98.26	98.21	98.08	98.44	98.29
16	92.53	92.73	92.84	92.70	92.63
17	94.55	94.72	94.47	94.58	94.67
18	87.13	87.00	87.24	86.91	86.79
19	91.84	92.19	93.18	92.41	91.89
20	95.72	96.39	96.56	96.13	96.38

Table A.8: Averaged predictions PINN, ANN, Linear Regression (red laser)

#	Real Output Y_i	ø PINN Output $\widehat{Y}_i$	PINN $(Y_i - \widehat{Y}_i)^2$	ø ANN Output $\widehat{Y}_i$	ANN $(Y_i - \widehat{Y}_i)^2$	ø Linear Regr. $\widehat{Y}_i$	Linear Regression $(Y_i - \widehat{Y}_i)^2$
1	87.5	86.76	0.54	86.89	0.38	86.75	0.57
2	96.75	96.77	0.0004	96.58	0.03	97.50	0.57
3	85.00	87.79	7.80	91.19	38.37	92.48	55.95
4	91.00	94.27	10.72	90.26	0.55	88.14	8.20
5	96.90	98.65	3.08	98.13	1.50	97.20	0.09
6	79.70	83.43	13.91	83.16	12.00	84.10	19.38
7	84.10	85.38	1.64	85.50	1.96	85.83	2.99

#	Real Output Y_i	ø PINN Output $\widehat{Y}_i$	PINN $(Y_i - \widehat{Y}_i)^2$	ø ANN Output $\widehat{Y}_i$	ANN $(Y_i - \widehat{Y}_i)^2$	ø Linear Regr. $\widehat{Y}_i$	Linear Regression $(Y_i - \widehat{Y}_i)^2$
8	99.50	98.74	0.58	98.41	1.18	98.71	0.63
9	86.50	85.76	0.54	86.33	0.03	86.51	0.0001
10	96.80	96.28	0.27	95.76	1.10	94.49	5.35
11	96.50	91.24	27.71	85.97	110.80	85.39	123.39
12	96.30	94.94	1.85	93.68	6.87	91.99	18.54
13	97.20	96.12	1.17	95.63	2.46	94.43	7.65
14	81.50	83.44	3.78	83.31	3.27	84.15	7.04
15	96.40	97.41	1.02	97.64	1.55	98.26	3.44
16	93.40	94.81	1.99	93.66	0.07	92.69	0.51
17	93.40	95.86	6.03	95.55	4.64	94.60	1.44
18	97.00	94.64	5.58	88.12	78.93	87.01	99.72
19	89.00	90.72	2.95	94.54	30.65	92.30	10.90
20	98.00	97.32	0.47	97.28	0.51	96.24	3.11
		MSE	**4.58**		**14.84**		**18.47**
		RMSE	**2.14**		**3.85**		**4.30**

Table A.9: Overview green laser data

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
1	190	1200	80	80	30	25	85.54
2	190	1200	60	80	30	25	90.01
3	190	1000	60	80	30	25	91.55
4	190	800	60	80	30	25	94.59
5	175	1000	80	80	30	25	88.64
6	175	1000	60	80	30	25	88.93
7	175	800	60	80	30	25	90.80

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
8	135	600	60	80	30	25	93.72
9	135	600	60	80	30	25	91.23
10	135	500	60	80	30	25	91.73
11	135	400	60	80	30	25	92.17
12	135	600	55	80	30	25	91.89
13	135	500	55	80	30	25	90.37
14	135	400	55	80	30	25	93.22
15	135	600	50	80	30	25	94.41
16	135	500	50	80	30	25	95.30
17	135	400	50	80	30	25	93.48
18	150	600	55	80	30	25	95.53
19	150	500	55	80	30	25	96.01
20	150	400	55	80	30	25	95.52
21	190	200	50	80	30	25	99.14
22	190	300	50	80	30	25	99.87
23	192	400	50	80	30	25	99.89
24	190	400	50	80	30	25	99.41
25	190	600	50	80	30	25	98.66
26	190	800	50	80	30	25	98.49
27	180	400	50	80	30	25	97.53
28	170	400	50	80	30	25	97.91
29	160	400	50	80	30	25	94.33
30	150	400	50	80	30	25	95.20
31	150	300	50	80	30	25	95.52
32	150	200	50	80	30	25	95.42
33	192	300	50	80	30	25	97.61

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
34	192	375	50	80	30	25	99.30
35	192	425	50	80	30	25	98.99
36	192	450	50	80	30	25	98.07
37	190	400	60	80	30	25	99.82
38	192	500	50	80	30	25	99.63
39	175	400	60	80	30	25	96.94
40	175	500	50	80	30	25	95.36
41	192	600	50	80	30	25	98.71
42	150	400	60	80	30	25	93.77
43	150	500	50	80	30	25	92.73
44	175	600	50	80	30	25	95.35
45	190	600	60	80	30	25	98.21
46	150	600	50	80	30	25	92.19
47	175	600	60	80	30	25	93.59
48	175	800	50	80	30	25	92.29
49	192	400	50	80	30	25	99.91
50	192	400	50	80	30	25	99.62
51	192	400	50	80	30	25	99.60
52	192	400	50	80	30	25	98.99

Table A.10: Random samples from Table A.9 for green laser validation set

#					
1	2	3	7	9	10
11	24	28	38	43	52

Table A.11: PINN predictions (green laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	91.00	91.04	91.39	90.90	90.62
2	95.89	97.28	96.95	95.82	96.81
3	92.49	92.18	93.24	92.12	92.57
4	93.60	93.13	93.55	93.10	93.75
5	99.29	99.24	99.41	99.21	99.34
6	95.16	94.98	94.76	94.65	94.87
7	99.17	99.12	99.25	99.10	99.25
8	90.60	91.41	91.31	90.60	91.22
9	92.15	92.32	91.80	91.03	91.97
10	92.98	92.88	92.34	92.55	92.42
11	87.83	89.34	88.56	87.86	89.53
12	99.22	99.16	99.33	99.10	99.26

Table A.12: ANN predictions (green laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	92.39	93.72	93.82	93.81	93.92
2	97.83	97.73	97.67	97.68	97.60
3	94.15	95.02	95.94	95.03	95.04
4	93.82	94.34	94.32	94.31	94.40
5	99.20	99.20	99.23	99.22	99.25
6	94.95	95.10	95.03	95.04	94.95
7	98.89	98.94	98.96	98.96	98.99
8	89.93	89.82	90.07	90.08	90.43
9	90.82	90.60	90.76	90.76	91.00
10	91.73	91.36	91.45	91.45	91.58

#	Output 1	Output 2	Output 3	Output 4	Output 5
11	88.59	88.81	89.23	89.13	89.68
12	99.11	99.10	99.11	99.11	99.13

Table A.13: Linear Regression predictions (green laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	92.81	92.12	91.75	91.78	91.03
2	97.10	97.13	97.08	97.09	97.25
3	94.04	93.44	93.25	93.28	93.39
4	93.91	93.48	93.37	93.43	93.62
5	99.07	99.01	99.12	99.08	99.07
6	94.69	94.75	94.48	94.53	94.86
7	98.46	98.35	98.36	98.33	98.34
8	91.55	91.37	91.17	91.32	91.77
9	92.16	92.03	91.92	92.07	92.51
10	92.77	92.68	92.67	92.83	93.24
11	90.44	89.24	89.41	89.57	89.71
12	98.89	98.84	98.93	98.90	98.91

Table A.14: Averaged predictions PINN, ANN, Linear Regression (green laser)

#	Real Output Y_i	ø PINN Output $\widehat{Y}_i$	PINN $(Y_i - \widehat{Y}_i)^2$	ø ANN Output $\widehat{Y}_i$	ANN $(Y_i - \widehat{Y}_i)^2$	ø Linear Regr. $\widehat{Y}_i$	Linear Regression $(Y_i - \widehat{Y}_i)^2$
1	90.01	90.99	0.96	93.53	12.40	91.90	3.56
2	97.91	96.55	1.85	97.70	0.04	97.13	0.61
3	91.55	92.52	0.94	95.04	12.15	93.48	3.72
4	90.80	93.43	6.90	94.24	11.82	93.56	7.63

#	Real Output Y_i	ø PINN Output $\widehat{Y}_i$	PINN $(Y_i - \widehat{Y}_i)^2$	ø ANN Output $\widehat{Y}_i$	ANN $(Y_i - \widehat{Y}_i)^2$	ø Linear Regr. $\widehat{Y}_i$	Linear Regression $(Y_i - \widehat{Y}_i)^2$
5	98.99	99.30	0.09	99.22	0.05	99.07	0.006
6	92.73	94.88	4.64	95.01	5.22	94.66	3.73
7	99.63	99.18	0.20	98.95	0.47	98.37	1.59
8	91.23	91.03	0.04	90.07	1.35	91.44	0.04
9	91.73	91.85	0.02	90.79	0.89	92.14	0.17
10	92.17	92.63	0.22	91.51	0.43	92.84	0.45
11	85.54	88.62	9.51	89.09	12.59	89.67	17.09
12	99.41	99.21	0.04	99.11	0.09	98.89	0.27
		MSE	**2.12**		**4.79**		**3.24**
		RMSE	**1.46**		**2.19**		**1.80**

Table A.15: Random samples from Table A.9 for reduced green laser validation set

#					
1	2	3	7	9	10
11	24	28	38	43	52
5	12	13	14	15	16
17	31	47	50		

Table A.16: PINN predictions (reduced green laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	91.32	91.43	90.28	88.31	91.74
2	96.67	96.47	97.20	96.84	96.68
3	92.69	92.64	91.17	91.07	93.24
4	93.72	93.47	92.58	92.61	94.27

#	Output 1	Output 2	Output 3	Output 4	Output 5
5	99.27	99.20	99.25	99.22	99.42
6	95.16	94.29	94.06	94.49	94.61
7	99.04	98.01	99.08	99.08	99.22
8	88.91	92.63	90.40	89.67	92.14
9	90.37	93.16	92.25	91.33	93.21
10	91.64	93.54	93.55	92.48	94.07
11	91.24	90.77	90.43	83.93	90.04
12	99.18	99.08	99.19	99.13	99.35
13	89.44	92.47	91.42	91.52	92.25
14	90.96	93.02	92.83	92.60	93.21
15	96.38	96.20	96.11	95.38	96.35
16	99.27	99.20	99.25	99.22	99.42
17	90.80	90.57	89.74	83.11	90.02
18	92.18	93.40	93.73	93.30	93.99
19	90.12	92.28	90.95	92.56	92.24
20	91.63	92.81	92.11	93.18	93.03
21	95.73	94.95	95.08	95.04	95.48
22	92.76	93.20	92.95	93.59	93.70

Table A.17: ANN predictions (reduced green laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	92.27	92.63	92.58	92.23	92.81
2	97.33	97.25	97.30	97.37	97.27
3	94.73	94.81	94.72	94.70	94.90
4	94.52	94.57	94.45	94.49	94.63
5	99.09	99.10	99.11	99.09	99.12

#	Output 1	Output 2	Output 3	Output 4	Output 5
6	93.60	93.59	93.65	93.60	93.64
7	98.67	98.66	98.69	98.67	98.69
8	89.55	90.26	90.13	89.41	90.37
9	91.02	91.40	91.23	90.99	91.47
10	92.38	92.53	92.33	92.26	92.56
11	92.77	93.18	92.65	92.56	93.32
12	98.97	98.98	99.00	98.98	99.00
13	89.42	90.12	90.13	89.32	90.26
14	90.89	91.27	91.22	90.80	91.35
15	96.49	96.45	96.35	96.47	96.48
16	99.09	99.10	99.11	99.09	99.12
17	92.53	92.93	92.36	92.32	93.03
18	92.26	92.40	92.32	92.18	92.44
19	89.29	90.00	90.13	89.23	90.13
20	90.77	91.14	91.22	90.72	91.24
21	95.74	95.60	95.64	95.77	95.61
22	92.15	92.27	92.32	92.11	92.33

Table A.18: Linear Regression predictions (reduced green laser)

#	Output 1	Output 2	Output 3	Output 4	Output 5
1	92.35	91.38	91.90	92.26	92.69
2	96.85	96.94	96.94	96.90	96.84
3	93.97	93.23	93.65	93.96	94.27
4	94.13	93.73	93.75	94.23	94.35
5	98.99	98.75	99.35	99.00	99.05
6	94.10	93.99	93.87	94.13	94.03

#	Output 1	Output 2	Output 3	Output 4	Output 5
7	98.18	97.82	98.48	98.15	98.25
8	91.86	91.95	91.10	92.10	91.92
9	92.67	92.88	91.98	92.95	92.72
10	93.48	93.81	92.85	93.80	93.51
11	92.42	91.88	91.42	92.77	93.07
12	98.80	98.56	99.13	98.81	98.84
13	91.85	91.83	91.22	91.97	91.83
14	92.66	92.76	92.10	92.82	92.62
15	95.75	95.59	95.50	95.93	95.94
16	98.99	98.75	99.35	99.00	99.05
17	92.58	92.38	91.52	93.04	93.15
18	93.47	93.69	92.97	93.67	93.41
19	91.83	91.70	91.34	91.84	91.73
20	92.64	92.63	92.22	92.69	92.52
21	95.72	95.85	95.61	95.83	95.62
22	93.45	93.56	93.09	93.54	93.32

Table A.19: Averaged predictions PINN, ANN, Linear Regression (reduced green laser)

#	Real Output Y_i	ø PINN Output $\widehat{Y}_i$	PINN $(Y_i - \widehat{Y}_i)^2$	ø ANN Output $\widehat{Y}_i$	ANN $(Y_i - \widehat{Y}_i)^2$	ø Linear Regr. $\widehat{Y}_i$	Linear Regression $(Y_i - \widehat{Y}_i)^2$
1	90.01	90.62	0.37	92.50	6.22	92.12	4.44
2	97.91	96.77	1.30	97.30	0.37	96.89	1.03
3	91.55	92.16	0.37	94.77	10.38	93.82	5.13
4	90.80	93.33	6.40	94.53	13.93	94.04	10.48
5	98.99	99.27	0.08	99.10	0.01	99.03	0.001
6	92.73	94.52	3.21	93.62	0.78	94.02	1.67

#	Real Output Y_i	ø PINN Output $\widehat{Y}_i$	PINN $(Y_i - \widehat{Y}_i)^2$	ø ANN Output $\widehat{Y}_i$	ANN $(Y_i - \widehat{Y}_i)^2$	ø Linear Regr. $\widehat{Y}_i$	Linear Regression $(Y_i - \widehat{Y}_i)^2$
7	99.63	98.89	0.55	98.68	0.91	98.18	2.11
8	91.23	90.75	0.23	89.94	1.65	91.79	0.31
9	91.73	92.06	0.11	91.22	0.26	92.64	0.83
10	92.17	93.06	0.78	92.41	0.06	93.49	1.74
11	85.54	89.28	14.00	92.90	54.11	92.31	45.86
12	99.41	99.19	0.05	98.99	0.18	98.83	0.34
13	91.89	91.42	0.22	89.85	4.16	91.74	0.02
14	90.37	92.52	4.64	91.11	0.54	92.59	4.94
15	93.59	96.08	6.22	96.45	8.17	95.74	4.63
16	99.62	99.27	0.12	99.10	0.26	99.03	0.35
17	88.64	88.85	0.04	92.63	15.95	92.53	15.16
18	93.22	93.32	0.01	92.32	0.81	93.44	0.05
19	94.41	91.63	7.73	89.76	21.66	91.67	7.41
20	95.30	92.55	7.55	91.02	18.34	92.54	7.62
21	95.52	95.26	0.07	95.67	0.02	95.73	0.04
22	93.48	93.24	0.06	92.24	1.55	93.39	0.008
		MSE	**2.46**		**7.29**		**5.19**
		RMSE	**1.57**		**2.70**		**2.28**

Table A.20: Poorly predicted data compared to similar training data with numbering from Table A.9 and sample number from Table A.19 in parenthesis

#	Laser Power	Scanning Speed	Hatch Distance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
7 (4)	**175**	**800**	**60**	**80**	**30**	**25**	**90.80**
4	190	800	60	80	30	25	94.59
47 (15)	**175**	**600**	**60**	**80**	**30**	**25**	**93.59**

#	Laser Power	Scanning Speed	Hatch Dis-tance	Laser Spot Diameter	Layer Thickness	Powder Size	Relative Density
8	135	600	60	80	30	25	93.72
45	190	600	60	80	30	25	98.21
15 (19)	**135**	**600**	**50**	**80**	**30**	**25**	**94.41**
46	150	600	50	80	30	25	92.19
44	175	600	50	80	30	25	95.35
25	190	600	50	80	30	25	98.66
41	192	600	50	80	30	25	98.71

Table A.21: Predictions PINN section 2 (reduced green laser)

#	Real Output Y_i	Section 2 Output $\widehat{Y}_i$	Section 2 $(Y_i - \widehat{Y}_i)^2$
1	90.01	93.66	13.32
2	97.91	96.27	2.69
3	91.55	93.77	4.93
4	90.80	93.94	9.86
5	98.99	98.89	0.01
6	92.73	93.72	0.98
7	99.63	98.88	0.56
8	91.23	93.73	6.25
9	91.73	93.88	4.62
10	92.17	94.19	4.08
11	85.54	93.69	66.42
12	99.41	98.89	0.27
13	91.89	93.64	3.06
14	90.37	93.71	11.16
15	93.59	95.13	2.37
16	99.62	98.89	0.53

#	Real Output Y_i	Section 2 Output $\widehat{Y}_i$	Section 2 $(Y_i - \widehat{Y}_i)^2$
17	88.64	93.88	27.46
18	93.22	93.84	0.38
19	94.41	93.62	0.62
20	95.30	93.65	2.72
21	95.52	94.20	1.74
22	93.48	93.70	0.05
		MSE	**7.46**
		RMSE	**2.73**

Appendix B

Table B.1: Excel files

Excel File Name	Content
ValidationData.xlsx	Validation data set for red laser data (20)
RegressionData.xlsx	Red laser data without validation data set (140)
Test_Data_Aconity.xlsx	Green laser data (52)
Data_Pure_Copper.xlsx	All red laser data (160)

Table B.2: Python code files not shown in detail

Code File Name	Reason
f_prediction_data.py	f_prediction_data_3.py with red laser data and red laser upper and lower bounds
green_data.py	Identical to f_prediction_data_3.py, different name for overview's sake as used for ANN instead of PINN
reduced_green_data.py	f_prediction_data_3.py with different amount of green laser data
config_2.py	config_3.py with red laser data and red laser upper and lower bounds
regressionmodel.py	regressionmodel_3.py with red laser data and red laser upper and lower bounds

Linear Regression

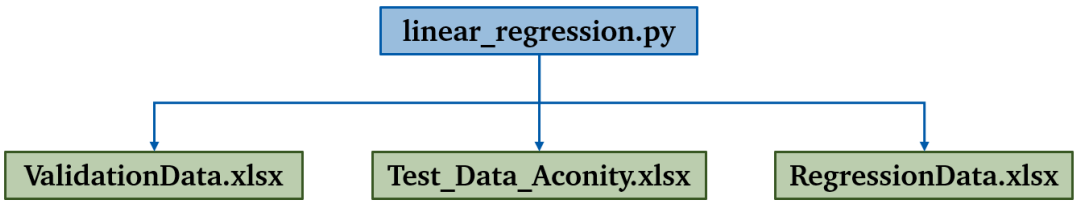


Figure B.1: Linear Regression code organization chart

Code B.1: Python file linear_regression.py

```
1   import numpy as np
2   import pandas as pd
3   from sklearn.model_selection import train_test_split
4   from sklearn.linear_model import LinearRegression
5
```

```
# set b = 0 for red laser data, set b = 1 for green laser data, set b
= 2 for reduced green laser data
b = 0
# load your dataset
if b == 0:
    excel_file_path = "RegressionData.xlsx"
    data = pd.read_excel(excel_file_path)
    data = data.iloc[:, 1:]  # ignore first column
elif b == 1:
    excel_file_path = "Test_Data_Aconity.xlsx"
    data = pd.read_excel(excel_file_path, skiprows=range(1,12),
nrows=40)
elif b == 2:
    excel_file_path = "Test_Data_Aconity.xlsx"
    data = pd.read_excel(excel_file_path, skiprows=range(1, 22),
nrows=30)

# separate input features (X) and target variable (y)
X = data.drop(columns=['rel_dens'])
y = data['rel_dens']

# split the dataset into a training set and a test set
# random_state can be changed for different regression
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=13)

# create linear regression object
model = LinearRegression()

# train the model using the training sets
model.fit(X_train, y_train)

# make predictions using the testing set
y_pred = model.predict(X_test)

if b == 0:
    excel_file_path2 = "ValidationData.xlsx"
    df = pd.read_excel(excel_file_path2, nrows=20)
    df = df.iloc[:, 1:]  # ignore first column
elif b == 1:
    df = pd.read_excel(excel_file_path, nrows=12)
elif b == 2:
    excel_file_path2 = "Test_Data_Aconity.xlsx"
    df = pd.read_excel(excel_file_path2, nrows=22)

# separate input features (X) and target variable (y)
inp_para = df.drop(columns=['rel_dens'])  # all columns except for
'rel_dens'
```

```
54    out_para = df['rel_dens']
55
56    # make predictions only using the input parameters
57    pred = model.predict(inp_para)
58
59    print(pred)
60
61    # save results in csv file
62    if b == 0:
63        csv_file_path = 'red_linear_output.csv'
64    elif b == 1:
65        csv_file_path = 'green_linear_output.csv'
66    elif b == 2:
67        csv_file_path = 'reduced_green_linear_output.csv'
68
69    np.savetxt(csv_file_path, pred, delimiter=',', fmt='%s')
```

Simple ANN Training

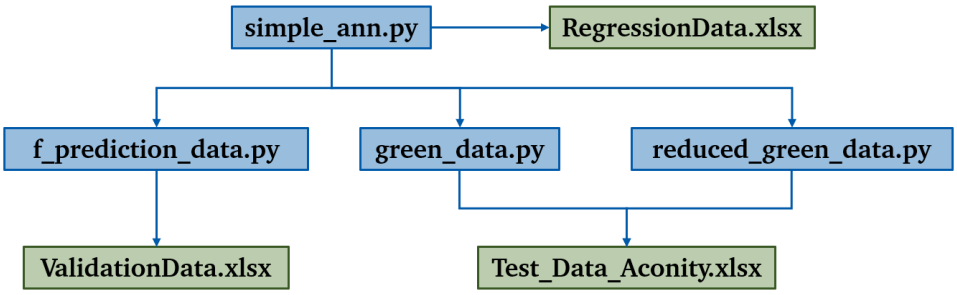


Figure B.2: Simple ANN code organization chart

Code B.2: Python file simple_ann.py

```
1   import tensorflow as tf
2   from tensorflow import keras
3   from tensorflow.keras import layers
4   import f_prediction_data
5   import pandas as pd
6   import numpy as np
7   import green_data
8   import reduced_green_data
9
10  # set b = 0 for red laser data, set b = 1 for green laser data, set b
11  = 2 for reduced green laser data
12  b = 1
13  # load your dataset
```

```
14  if b == 0:
15      excel_file_path = "RegressionData.xlsx"
16      df = pd.read_excel(excel_file_path)
17      df = df.iloc[:, 1:]  # ignore first column as that includes the
18  title
19  elif b == 1:
20      excel_file_path = "Test_Data_Aconity.xlsx"
21      df = pd.read_excel(excel_file_path, skiprows=range(1,12),
22  nrows=40)
23  elif b == 2:
24      excel_file_path = "Test_Data_Aconity.xlsx"
25      df = pd.read_excel(excel_file_path, skiprows=range(1, 22),
26  nrows=30)
27
28  # load the input parameters
29  lp = df.loc[:, 'laser_power'].values
30  ss = df.loc[:, 'scan_speed'].values
31  hd = df.loc[:, 'hatch_dist'].values
32  lsd = df.loc[:, 'laser_sd'].values
33  lt = df.loc[:, 'layer_thick'].values
34  ps = df.loc[:, 'powd_size'].values
35
36  # normalize input values separately before combining
37  if b == 0:
38      # upper bound: 1200, lower bound: 100 [W]
39      norm_lp = (lp - 100) / (1200 - 100)
40      # upper bound: 1200, lower bound: 50 [mm/s]
41      norm_ss = (ss - 50) / (1200 - 50)
42      # upper bound: 120, lower bound: 10 [micro m]
43      norm_hd = (hd - 10) / (120 - 10)
44      # upper bound: 100, lower bound: 20 [micro m]
45      norm_lsd = (lsd - 20) / (100 - 20)
46      # upper bound: 55, lower bound: 10 [micro m]
47      norm_lt = (lt - 10) / (55 - 10)
48      # upper bound: 50, lower bound: 15 [micro m]
49      norm_ps = (ps - 15) / (50 - 15)
50  elif b == 1 or b == 2:
51      # upper bound: 200, lower bound: 100 [W]
52      norm_lp = (lp - 100) / (200 - 100)
53      # upper bound: 1200, lower bound: 50 [mm/s]
54      norm_ss = (ss - 50) / (1200 - 50)
55      # upper bound: 70, lower bound: 30 [micro m]
56      norm_hd = (hd - 30) / (70 - 30)
57      # upper bound: 80, lower bound: 40 [micro m]
58      norm_lsd = (lsd - 40) / (80 - 40)
59      # upper bound: 40, lower bound: 20 [micro m]
60      norm_lt = (lt - 20) / (40 - 20)
61      # upper bound: 30, lower bound: 20 [micro m]
```

```
62      norm_ps = (ps - 20) / (30 - 20)
63
64  # combine all normalized input parameters in one matrix
65  inp = np.column_stack((norm_lp, norm_ss, norm_hd, norm_lsd, norm_lt,
66  norm_ps))
67
68  dens = df.loc[:, 'rel_dens'].values
69
70  # normalize values for relative density using logarithmic transfor-
71  mation [0,1]
72  log_trans_dens = np.log(101 - dens.astype(float))
73  lower_bound = np.log(1)
74  upper_bound = np.log(31.0)
75  inv_outp = (log_trans_dens - lower_bound) / (upper_bound -
76  lower_bound)
77  outp = 1 - inv_outp
78  X = inp
79  y = outp
80
81  # define architecture of the model
82  model = keras.Sequential([
83      layers.Dense(32, activation='sigmoid', input_shape=(6,)),
84      layers.Dense(16, activation='sigmoid'),
85      layers.Dense(8, activation='sigmoid'),
86      layers.Dense(1)
87  ])
88
89  # compile the model
90  optimizer = keras.optimizers.Adam(learning_rate=0.01)
91  model.compile(optimizer=optimizer, loss='mean_squared_error')
92
93  # train the model
94  if b == 0:
95      model.fit(X, y, epochs=100, batch_size=32, validation_split=0.2)
96  elif b == 1 or b == 2:
97      model.fit(X, y, epochs=200, batch_size=32, validation_split=0.2)
98  # adjust epochs and batch_size as needed
99
100
101 # load the prediction data from green_data, only use the input data
102 if b == 0:
103     [inp_f, out_f] = f_prediction_data.load_prediction_data(1)
104 elif b == 1:
105     [inp_f, out_f] = green_data.load_prediction_data(1)
106 elif b == 2:
107     [inp_f, out_f] = reduced_green_data.load_prediction_data(1)
108
109 new_inputs = inp_f
```

```
predictions = model.predict(new_inputs)

# reverse the normalization
lower_bound = np.log(1)
upper_bound = np.log(31.0)
norm_log_trans_dens = 1 - predictions
log_trans_dens = norm_log_trans_dens * (upper_bound - lower_bound) +
lower_bound

# reverse the logarithmic transformation
pred_dens = 101 - np.exp(log_trans_dens)

# print the predictions
print(pred_dens)


# save results in csv file
if b == 0:
    csv_file_path = 'red_ann_output.csv'
elif b == 1:
    csv_file_path = 'green_ann_output.csv'
elif b == 2:
    csv_file_path = 'reduced_green_ann_output.csv'

np.savetxt(csv_file_path, pred_dens, delimiter=',', fmt='%s')
```

PINN Training

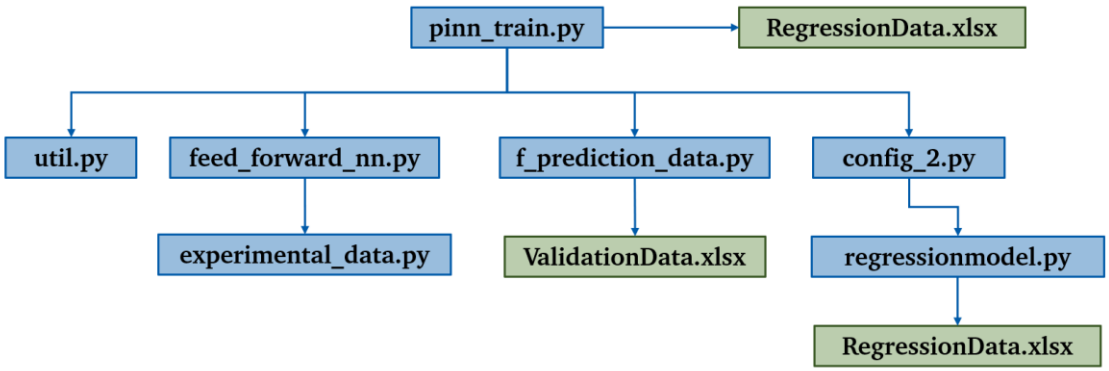


Figure B.3: PINN code organization chart (red laser data)

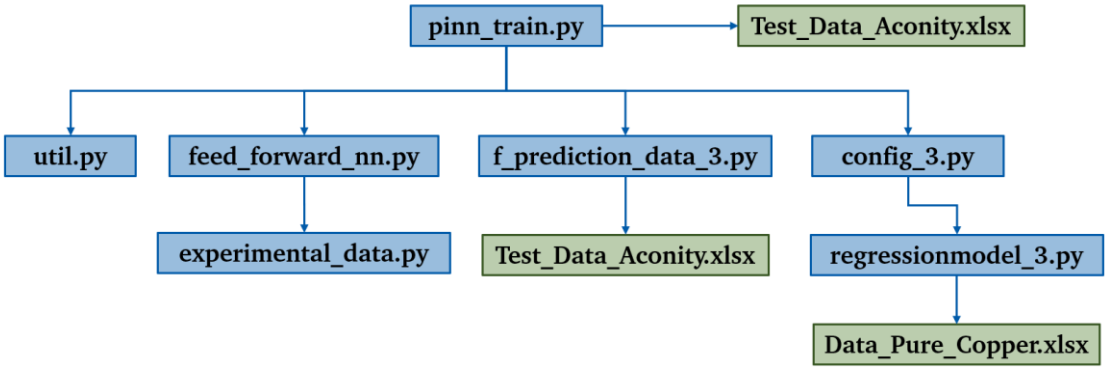


Figure B.4: PINN code organization chart (green laser data)

pinn.train.py

```
from files import feed_forward_nn as ffnn
import config_3 as config
from files import util as ut
import f_prediction_data_3
import numpy as np

# load configuration
config = config.pinn_config()

# initiate model
f_model = ffnn.feed_forward_nn(config)
# train model with physical data
f_model.model_fit_with_physical_data()
print("mess data:")
f_model.model_fit_with_mess_data()
# save model in model folder
f_model.save_as()

# load prediction data (only inp_f will be used)
[inp_f, out_f] = f_prediction_data_3.load_prediction_data(1)
# use model for prediction
pred = f_model.pred_model(inp_f, 1)

# reverse the normalization
lower_bound = np.log(1)
upper_bound = np.log(31.0)
norm_log_trans_dens = 1 - pred
log_trans_dens = norm_log_trans_dens * (upper_bound - lower_bound) +
lower_bound

# reverse the logarithmic transformation
pred_dens = 101 - np.exp(log_trans_dens)
```

```
print(pred_dens)

# save predictions in csv file
csv_file_path = 'green_pinn_output.csv'

np.savetxt(csv_file_path, pred_dens, delimiter=',', fmt='%s')
```

Code B.3: Python file config_3.py

```
from tensorflow import keras
import numpy as np
import scipy.interpolate as inter
import os
import pandas as pd
from regressionmodel_3 import regression

excel_file_path_2 = "Test_Data_Aconity.xlsx"
nc = pd.read_excel(excel_file_path_2)

def global_config():
    # feel free to change here
    config = {
        'len_lhs': 32,  # number of samples which are used to sample
the function space (latin hypercube sampling)
        'anz_b': 2,  # number of total cases
        'anz_b_phy': 1,  # number of cases for pre-training
        'size_b': 5,  # size of compressed behaviour vector
        'anz_inp_param': 6,  # total number of input parameters
        'anz_out_param': 1,  # total number of output parameters
        'save_to': 'temp_00', # path where everything is stored
        'test_split': 0 # how much data is not used in training
    }

    config['name'] = 'model' #defines main directory


    # where saved files are stored
    config['save_to_csv'] = os.path.join(config['save_to'], 'CSV')
    config['save_to_pinn'] = os.path.join(config['save_to'],
'pinn_models')
    config['save_to_sdnn'] = os.path.join(config['save_to'],
'sdnn_model')
    config['save_to_ae'] = os.path.join(config['save_to'], 'ae_mod-
els')
    config['save_to_rnn'] = os.path.join(config['save_to'], 'rnn_mod-
els')
    config['save_to_f'] = os.path.join(config['save_to'], 'f_models')
```

```
39      config['save_to_models'] = os.path.join(config['save_to'], 'mod-
40  els')
41
42
43      def set_architecture(config):
44
45          # architecture for PINN
46          if config['archi_type'] == 0:
47              inp_shape = (config['inp_shape'],)
48
49
50              dropout = 0.05
51              # model should have some level of complexity,
52              model = keras.Sequential()
53
54              model.add(keras.layers.Dense(32, activation='sigmoid',in-
55  put_shape = inp_shape ))
56              model.add(keras.layers.Dropout(dropout))
57              model.add(keras.layers.Dense(16, activation='sigmoid'))
58              model.add(keras.layers.Dropout(dropout))
59              model.add(keras.layers.Dense(8, activation='sigmoid'))
60              model.add(keras.layers.Dropout(dropout))
61              model.add(keras.layers.Dense(1, activation='sigmoid'))
62
63              optimizer = keras.optimizers.Adam(learning_rate=0.01)
64              model.compile(optimizer=optimizer,loss=keras.met-
65  rics.mean_squared_error)
66
67              return model
68
69
70      # this is the synthetic red laser data used for pretraining
71      def generate_physical_training_data(b,inp):
72          b = 0
73          x_anz = inp.shape[0]
74
75          # load unnormalized synthetic data from the regression model
76          [input_parameter, output_parameter] = regression(x_anz)
77
78          # create variables for each column
79          lp, ss, hd, lsd, lt, ps = input_parameter.T
80
81          # normalize input values separately before combining
82          # upper bound: 1200, lower bound: 100 [W]
83          norm_lp = (lp - 100) / (1200 - 100)
84          # upper bound: 1200, lower bound: 50 [mm/s]
85          norm_ss = (ss - 50) / (1200 - 50)
86          # upper bound: 120, lower bound: 10 [micro m]
```

```
87          norm_hd = (hd - 10) / (120 - 10)
88          # upper bound: 100, lower bound: 20 [micro m]
89          norm_lsd = (lsd - 20) / (100 - 20)
90          # upper bound: 55, lower bound: 10 [micro m]
91          norm_lt = (lt - 10) / (55 - 10)
92          # upper bound: 50, lower bound: 15 [micro m]
93          norm_ps = (ps - 15) / (50 - 15)
94
95          # combine all normalized input parameters in one matrix
96          inp = np.column_stack((norm_lp, norm_ss, norm_hd, norm_lsd,
97  norm_lt, norm_ps))
98
99          # normalize values for relative density using logarithmic
100 transformation [0,1]
101         log_trans_dens = np.log(101 - output_parameter.astype(float))
102         lower_bound = np.log(1)
103         upper_bound = np.log(31.0)
104         inv_outp = (log_trans_dens - lower_bound) / (upper_bound -
105 lower_bound)
106         outp = 1 - inv_outp
107
108         return [inp, outp]
109
110
111     # this is the green laser data used for training
112     def load_mess_data(b):
113
114         # load data from excel file
115         x = 12
116         y = 51
117
118         # laser_power
119         lp = nc.loc[x:y, 'laser_power'].values
120         # upper bound: 200, lower bound: 100 [W]
121         norm_lp = (lp - 100) / (200 - 100)
122
123         # scan_speed
124         ss = nc.loc[x:y, 'scan_speed'].values
125         # upper bound: 1200, lower bound: 50 [mm/s]
126         norm_ss = (ss - 50) / (1200 - 50)
127
128         # hatch_dist
129         hd = nc.loc[x:y, 'hatch_dist'].values
130         # upper bound: 70, lower bound: 30 [micro m]
131         norm_hd = (hd - 30) / (70 - 30)
132
133         # laser_sd
134         lsd = nc.loc[x:y, 'laser_sd'].values
```

```
135             # upper bound: 80, lower bound: 40 [micro m]
136             norm_lsd = (lsd - 40) / (80 - 40)
137 
138             # layer_thick
139             lt = nc.loc[x:y, 'layer_thick'].values
140             # upper bound: 40, lower bound: 20 [micro m]
141             norm_lt = (lt - 20) / (40 - 20)
142 
143             # powd_size
144             ps = nc.loc[x:y, 'powd_size'].values
145             # upper bound: 30, lower bound: 20 [micro m]
146             norm_ps = (ps - 20) / (30 - 20)
147 
148             # combine all normalized input parameters in one matrix
149             inp_matrix = np.column_stack((norm_lp, norm_ss, norm_hd,
150 norm_lsd, norm_lt, norm_ps))
151 
152             # goal value: rel_dens
153             dens = nc.loc[x:y, 'rel_dens'].values
154 
155             # normalize values for relative density using logarithmic
156 transformation
157             log_trans_dens = np.log(101 - dens.astype(float))
158             lower_bound = np.log(1)
159             upper_bound = np.log(31.0)
160             norm_log_trans_dens = (log_trans_dens - lower_bound) / (up-
161 per_bound - lower_bound)
162 
163             x_data = inp_matrix
164             y_data = 1 - norm_log_trans_dens
165             return [x_data, y_data]
166 
167     config["set_architecture"] = set_architecture
168     config["generate_physical_training_data"] = generate_physi-
169 cal_training_data
170     config["load_mess_data"] = load_mess_data
171     return config
172 
173 # hyperparameters for PINN model
174 def pinn_config():
175     config = global_config() # global config for all configs
176     config['save_to_models'] = config['save_to_pinn']
177     config['name'] = 'pinn'
178     config['inp_shape'] = config['anz_inp_param'] + config['size_b']
179     config['pinn_samples'] = 10000 # amount of synthetic data
180     config['pinn_gen_new'] = False #if new samples are to be gener-
181 ated. if there are no, new samples are generated
182 
```

```
183         config['pinn_overwrite_old'] = False # whether new samples should
184     be added to existing samples or overwritten
185 
186         # hyperparameter for training
187         config['validation_split'] = 0.2
188         config['batch_size'] = 32
189         config['epochs'] = 30
190         config['verbose'] = 1
191         config['multistep_fit'] = 2
192         config['archi_type'] = 0
193         return config
```

Code B.4: Python file regressionmodel_3.py

```
1   import pandas as pd
2   from sklearn.model_selection import train_test_split
3   from sklearn.ensemble import RandomForestRegressor
4   from sklearn.metrics import mean_squared_error, r2_score
5   from sklearn.preprocessing import MinMaxScaler
6   import numpy as np
7 
8   def regression(n):
9       # load red laser data from Excel file
10      excel_file_path = 'Data_Pure_Copper.xlsx'
11      df = pd.read_excel(excel_file_path)
12      df = df.iloc[:, 1:]  # ignore first column as it contains title
13 
14      # separate input parameters X and output y
15      X = df.drop(columns=['rel_dens'])
16      y = df['rel_dens']
17 
18      # define the desired range for each input parameter
19      parameter_ranges = {
20          'laser_power': (100, 1200),
21          'laser_sd': (20, 100),
22          'scan_speed': (50, 1200),
23          'hatch_dist': (10, 120),
24          'powd_size': (15, 50),
25          'layer_thick': (10, 55),
26      }
27 
28      # scale input parameters
29      scaler = MinMaxScaler(feature_range=(0, 1), copy=True)
30 
31      # apply scaling to the input parameters within the specified
32  ranges
33      for parameter, (min_value, max_value) in parame-
34  ter_ranges.items():
```

```
35          X[parameter] = scaler.fit_transform(X[[parameter]])
36          X[parameter] = X[parameter] * (max_value - min_value) +
37  min_value
38
39      # split the data into training and testing sets
40      X_train, X_test, y_train, y_test = train_test_split(X, y,
41  test_size=0.2, random_state=42)
42
43      # random forest regressor model
44      model = RandomForestRegressor(n_estimators=100, random_state=12)
45
46      # train the model
47      model.fit(X_train, y_train)
48
49      # make predictions on the test set
50      y_pred = model.predict(X_test)
51
52      # evaluate the model
53      mse = mean_squared_error(y_test, y_pred)
54      r2 = r2_score(y_test, y_pred)
55
56      print(f'Mean Squared Error: {mse:.2f}')
57      print(f'R-squared: {r2:.2f}')
58
59      # use the trained model to generate new data points
60      new_data_points = pd.DataFrame(np.random.rand(n, len(X.columns)),
61  columns=X.columns)
62
63      # apply scaling to the new data points within the specified
64  ranges
65      for parameter, (min_value, max_value) in parame-
66  ter_ranges.items():
67          new_data_points[parameter] = scaler.fit_trans-
68  form(new_data_points[[parameter]])
69          new_data_points[parameter] = new_data_points[parameter] *
70  (max_value - min_value) + min_value
71
72      # Make predictions on the new data points
73      predicted_rel_densities = model.predict(new_data_points)
74
75      # generated data points and their predicted relative densities
76      generated_data = pd.concat([new_data_points, pd.Series(pre-
77  dicted_rel_densities, name='predicted_rel_dens')], axis=1)
78      outp = generated_data['predicted_rel_dens']
79      inp = np.column_stack((generated_data['laser_power'], gener-
80  ated_data['scan_speed'], generated_data['hatch_dist'], gener-
81  ated_data['laser_sd'], generated_data['layer_thick'], gener-
82  ated_data['powd_size']))
```

```
83
84          return[inp, outp]
85
```

Code B.5: Python file f_prediction_data_3.py

```
1   import numpy as np
2   import pandas as pd
3
4   excel_file_path1 = "Test_Data_Aconity.xlsx"
5   df = pd.read_excel(excel_file_path1)
6
7
8   def load_prediction_data(n):
9       if n == 0:
10          x = 12
11          y = 51
12      elif n == 1:
13          x = 0
14          y = 11
15
16      # laser_power
17      lp = df.loc[x:y, 'laser_power'].values
18      # upper bound: 200, lower bound: 100 [W]
19      norm_lp = (lp - 100) / (200 - 100)
20
21      # scan_speed
22      ss = df.loc[x:y, 'scan_speed'].values
23      # upper bound: 1200, lower bound: 50 [mm/s]
24      norm_ss = (ss - 50) / (1200 - 50)
25
26      # hatch_dist
27      hd = df.loc[x:y, 'hatch_dist'].values
28      # upper bound: 70, lower bound: 30 [micro m]
29      norm_hd = (hd - 30) / (70 - 30)
30
31      # laser_sd
32      lsd = df.loc[x:y, 'laser_sd'].values
33      # upper bound: 80, lower bound: 40 [micro m]
34      norm_lsd = (lsd - 40) / (80 - 40)
35
36      # layer_thick
37      lt = df.loc[x:y, 'layer_thick'].values
38      # upper bound: 40, lower bound: 20 [micro m]
39      norm_lt = (lt - 20) / (40 - 20)
40
41      # powd_size
```

```
42        ps = df.loc[x:y, 'powd_size'].values
43        # upper bound: 30, lower bound: 20 [micro m]
44        norm_ps = (ps - 20) / (30 - 20)
45
46        # combine all normalized input parameters in one matrix
47        inp_matrix = np.column_stack((norm_lp, norm_ss, norm_hd,
48    norm_lsd, norm_lt, norm_ps))
49
50        # goal value: rel_dens
51        dens = df.loc[x:y, 'rel_dens'].values
52        log_trans_dens = np.log(101 - dens.astype(float))
53        lower_bound = np.log(1)
54        upper_bound = np.log(31.0)
55        inv_outp = (log_trans_dens - lower_bound) / (upper_bound -
56    lower_bound)
57        outp = 1 - inv_outp
58
59        x_data = inp_matrix
60        y_data = outp
61
62        return [x_data, y_data]
```

Code B.6: Python file feed_forward_nn.py

```
1   from tensorflow import keras
2   from keras import backend as K
3   from files import experimental_data as exd
4   import numpy as np
5   import matplotlib.pyplot as plt
6   from files import util
7   from tensorflow.keras.callbacks import LearningRateScheduler
8   import tensorflow as tf
9
10
11  class feed_forward_nn:
12      # define the architecture and all parameters
13      archi_type = None
14      # training parameters
15      dropout = None
16      validation_split = None
17      batch_size = None
18      epochs = None
19      verbose = None
20      ae_dimi = None
21      len_lhs = None
22      anz_b = None
23      anz_b_phy = None
```

```
24      anz_param = None
25      config = None
26      #actual keras model
27      model = 0
28      encoder =None
29      decoder =None
30
31      # initiate the init function that initiates model, parameters de-
32  fined in config file
33      def __init__(self, config, model = None ):
34
35          self.archi_type = config["archi_type"]
36
37          self.validation_split = config['validation_split'] if 'vali-
38  dation_split' in config else 0.2
39          self.batch_size = config['batch_size'] if 'batch_size' in
40  config else 256
41          self.epochs = config['epochs'] if 'epochs' in config else 100
42          self.verbose = config['verbose'] if 'verbose' in config else
43  0
44          self.ae_dimi = config['ae_dimi'] if 'ae_dimi' in config else
45  2
46          self.len_lhs = config['len_lhs']
47          self.anz_b = config['anz_b']
48          self.anz_b_phy = config['anz_b_phy']
49          self.anz_param = config['anz_inp_param']
50
51          self.config = config
52          if model is None:
53              self.model = config["set_architecture"](config)
54
55          if config['name']=='ae':
56              self.encoder = self.get_compressed_layer_feature_extrac-
57  tor()
58              self.decoder = self.get_decompressed_layer_feature_ex-
59  tractor()
60
61
62      # train model
63      def train(self, input_data, output_data):
64          if self.config['name'] == 'pinn' or self.config['name'] ==
65  'sdnn':
66              n = int(input_data.shape[0]*(1-self.con-
67  fig['test_split']))
68
69              if self.config["anz_inp_param"] == 1:
70                  input_data = input_data[:n]
71              else:
```

```
72                  input_data = input_data[:n, :]
73
74              if self.config["anz_out_param"] == 1:
75                  output_data = output_data[:n]
76              else:
77                  output_data = output_data[:n, :]
78          if False:
79              # generate a permutation index
80              permutation_index = np.random.permutation(in-
81  put_data.shape[0])
82
83              # shuffle both matrices using the same permutation index
84              output_data = output_data[permutation_index]
85              input_data = input_data[permutation_index]
86
87          for step in range(self.config['multistep_fit']):
88              dimi = int(2**step)
89              bt_s = int(self.batch_size*dimi)
90              epo = int(self.epochs/dimi)
91              hist = self.model.fit(input_data, output_data, valida-
92  tion_split = self.validation_split,
93                   batch_size=bt_s, epochs=epo, verbose=self.verbose)
94          return hist
95
96      def pred_model(self, input_data, b=None, model=None):
97          if model is None:
98              model = self.model
99
100         if b is None:
101             return model.predict(input_data, verbose=self.con-
102 fig["verbose"])
103         else:
104             ex = exd.experimental_data(self.config)
105
106             input_data = ex.make_bin_data(input_data,b)
107
108             return model.predict(input_data)
109
110
111     # save model
112     def save_as(self, file_name=None):
113
114         utils = util.util(self.config)
115         utils.save_model(self.model,model_name = file_name)
116
117
118     # load model
119     def load(self, file_index=-1, model_name=None):
```

```
120
121         utils = util.util(self.config)
122         self.model = utils.load_model(file_index = file_in-
123 dex,model_name = model_name)
124
125
126     # train model with synthetic data from regression
127     def model_fit_with_physical_data(self):
128
129         ex_d = exd.experimental_data(self.config)
130         [pretrain_x,pretrain_y] = ex_d.generate_physical_data()
131
132         self.train(pretrain_x,pretrain_y)
133
134
135     # train model with measured experimental data
136     def model_fit_with_mess_data(self):
137
138         ex_d = exd.experimental_data(self.config)
139         [train_x,train_y] = ex_d.load_messurement_data()
140
141
142         return self.train(train_x, train_y)
```

Code B.7: Python file util.py

```
1  import os
2  import time
3  import numpy as np
4  from pyDOE import lhs
5  from tensorflow import keras
6  from files import feed_forward_nn as ffnn
7  from files import experimental_data as ed
8
9  class util:
10     config = None
11
12     def __init__(self, config):
13         self.config = config
14
15     #load model
16     def load_model(self, keyword=None, path=None, file_index=-1,
17 model_name=None):
18
19         if keyword is None:
20             keyword=self.config['name']
21
```

```
22          if path is None:
23              path=self.config['save_to_models']
24
25          if model_name is None:
26              model_name_list = self.get_load_list(path, keyword,
27  'json')
28              model_name = model_name_list[file_index]
29
30          model_name = os.path.join(path,model_name)
31          json_file = open(model_name+'.json', 'r')
32          loaded_model_json = json_file.read()
33          json_file.close()
34          loaded_model = keras.mod-
35  els.model_from_json(loaded_model_json)
36
37          #load weights into new model
38          loaded_model.load_weights(model_name+'.h5')
39          return loaded_model
40
41
42      # save model
43      def save_model(self, model, path = None, model_name=None):
44
45          if path is None:
46              path=self.config['save_to_models']
47
48          if model_name is None:
49              model_name = self.config['name'] + "" +
50  str(int(time.time()*1000))
51
52          model_path_name = os.path.join(path, model_name)
53          model_json = model.to_json()
54          if not os.path.exists(path):
55              os.makedirs(path)
56          with open(model_path_name + '.json', 'w') as json_file:
57              json_file.write(model_json)
58
59          model.save_weights(model_path_name+'.h5')
60          print('Saved ' + self.config['name'] + '-model to disk suc-
61  cessfully')
62          return model_name
63
64
65      # load array from csv file
66      def load_array(self, keyword=None, path=None, file_index=-1, ar-
67  ray_name=None):
68
69          if path is None:
```

```
70              path=self.config['save_to_csv']
71
72          if array_name is None:
73              if keyword is None:
74                  keyword='x_lhs'
75              array_name_list = self.get_load_list(path, keyword,
76  'csv')
77
78              if file_index < len(array_name_list) and array_name_list:
79                  array_name = array_name_list[file_index]
80                  array_name = os.path.join(path,array_name+'.csv')
81                  loaded_array = np.genfromtxt(array_name, delim-
82  iter=',')
83                  return loaded_array
84              else:
85                  array_name_list = self.get_load_list(path, keyword,
86  'npy')
87                  if file_index < len(array_name_list) and ar-
88  ray_name_list:
89                      array_name = array_name_list[file_index]
90                      array_name = os.path.join(path,array_name+'.npy')
91                      loaded_array = np.load(array_name)
92                      return loaded_array
93                  else:
94                      return None
95          else:
96              array_name_tmp = os.path.join(path,array_name+'.csv')
97              if os.path.exists(array_name_tmp):
98                  loaded_array = np.genfromtxt(array_name_tmp, delim-
99  iter=',')
100             else:
101                 array_name = os.path.join(path,array_name+'.npy')
102                 loaded_array = np.load(array_name)
103             return loaded_array
104
105
106     # save array as csv file
107     def save_array(self,array, keyword, path=None , array_name=None):
108
109         if array_name is None:
110             array_name = keyword + "" + str(int(time.time()*1000))
111
112         if path is None:
113             path = self.config['save_to_csv']
114
115         if not os.path.exists(path):
116             os.makedirs(path)
117         if len(array.shape)<3:
```

```
118                 array_path_name = os.path.join(path, array_name + '.csv')
119                 np.savetxt(array_path_name, array, delimiter=',',
120   fmt='%e')
121             else:
122                 array_path_name = os.path.join(path, array_name)
123                 np.save(array_path_name,array)
124             #print('Saved ' + keyword + ' to disk successfully')
125             return array_name
126
127
128         # load list saved in directory
129         def get_load_list(self, path, keyword, file_type):
130
131             if not os.path.exists(path):
132                 os.makedirs(path)
133
134             files = [f for f in os.listdir(path) if f.endswith(file_type)
135   and f.startswith(keyword)]
136             files_sorted = sorted(files)
137             files_without_extension = [os.path.splitext(file)[0] for file
138   in files_sorted]
139
140             return files_without_extension
141
142
143         def get_x_lhs(self):
144             x_lhs = self.load_array('x_lhs')
145             if x_lhs is None:
146                 # generate latin hypercube sampling
147                 random_seed = 123  # set the random seed for repro-
148   ducebility of x_lhs
149                 np.random.seed(random_seed)
150                 x_lhs = lhs(self.config['anz_inp_param'], sam-
151   ples=self.config['len_lhs'])
152
153                 self.save_array(x_lhs, 'x_lhs')
154
155             return x_lhs
```

Code B.8: Python file experimental_data.py

```
1   import numpy as np
2   import os
3
4   class experimental_data:
5       anz_param = None
6       anz_b = None
```

```
7       len_lhs = None
8       config = None
9
10      # initiate constructor function
11      def __init__(self, config):
12          self.anz_param = config['anz_inp_param']
13          self.anz_b = config['anz_b']
14          self.len_lhs = config['len_lhs']
15          self.config = config
16
17
18      # generate synthetic data from regression
19      def generate_physical_data(self, b=None, inp = None, x_anz =
20  None):
21          # create parameters to teach
22          # number of values
23          if b is None:
24              inp_path = os.path.join(self.config['save_to_csv'],
25  'inp_phy.csv')
26              out_path = os.path.join(self.config['save_to_csv'],
27  'out_phy.csv')
28              gen_new = os.path.exists(inp_path) and os.path.ex-
29  ists(out_path)
30              gen_new = not gen_new
31
32              if self.config['pinn_gen_new'] or gen_new:
33                  inp_out = 0
34                  for b in range(self.config['anz_b_phy']):
35                      [inp_tmp,outp_tmp] = self.generate_physi-
36  cal_data(b,inp, x_anz)
37                      anz = inp_tmp.shape
38
39                      if b == 0:
40                          inp_out = inp_tmp
41                          outp_out= outp_tmp
42                      else:
43                          inp_out = np.concatenate((inp_out,inp_tmp))
44                          outp_out = np.concate-
45  nate((outp_out,outp_tmp))
46                  if not (self.config["pinn_overwrite_old"] or
47  gen_new):
48                      inp_phy = np.genfromtxt(inp_path, delimiter=',')
49                      outp_phy = np.genfromtxt(out_path, delimiter=',')
50                      inp_out = np.concatenate((inp_phy,inp_out))
51                      outp_out = np.concatenate((outp_phy,outp_out))
52
53                  if not os.path.exists(self.config['save_to_csv']):
54                      os.makedirs(self.config['save_to_csv'])
```

```
55                      np.savetxt(inp_path, inp_out, delimiter=',',
56  fmt='%f')
57                      np.savetxt(out_path, outp_out, delimiter=',',
58  fmt='%f')
59
60              else:
61                  inp_out = np.genfromtxt(inp_path, delimiter=',')
62                  outp_out = np.genfromtxt(out_path, delimiter=',')
63              return [inp_out, outp_out]
64
65          if inp is None:
66              if x_anz is None:
67                  x_anz = self.config["pinn_samples"]
68              inp = np.random.rand(x_anz,self.anz_param)
69              outp = np.zeros((x_anz,self.config['anz_out_param']))
70          else:
71              anz = inp.shape
72              x_anz = anz[0]
73              outp = np.zeros(x_anz)
74
75
76          [inp,outp] = self.config["generate_physical_train-
77  ing_data"](b,inp)
78          b_bin = self.int_2_bin_arr(b)
79          inp_size = inp.shape
80          x_anz =inp_size[0]
81          b_bin = np.array([b_bin for tmp in range(x_anz)])
82          inp = np.hstack((b_bin,inp))
83          return [inp,outp]
84
85      # load measured experimental data
86      def load_messurement_data(self):
87          for ii in range(self.config['anz_b_phy'], self.con-
88  fig['anz_b']):
89              [inp_tmp, outp_tmp] = self.config["load_mess_data"](ii)
90              b_bin = self.int_2_bin_arr(ii)
91              inp_size = inp_tmp.shape
92              x_anz = inp_size[0]
93              b_bin = np.array([b_bin for tmp in range(x_anz)])
94              inp_tmp = np.hstack((b_bin, inp_tmp))
95              if ii == self.config['anz_b_phy']:
96                  inp_out = inp_tmp
97                  outp_out= outp_tmp
98              else:
99                  inp_out = np.concatenate((inp_out,inp_tmp))
100                 outp_out = np.concatenate((outp_out,outp_tmp))
101
102         return [inp_out, outp_out]
```

```
103
104
105         # turn integer into numpy array
106         def int_2_bin_arr(self, integ, length = None):
107
108             if length is None:
109                 length = self.config['size_b']
110             integ = int(integ)
111             tmp_str = format(integ, "b")
112             bin_arr = [c for c in tmp_str]
113             bin_arr = np.array(bin_arr)
114             bin_arr = bin_arr.astype(np.float64)
115
116             while len(bin_arr)<length:
117                 bin_arr = np.insert(bin_arr,0,0)
118             while len(bin_arr) > length:
119                 bin_arr = np.delete(bin_arr,0)
120             return bin_arr
121
122
123         # turn input_data into correct shape
124         def make_bin_data(self, input_data, b):
125             if input_data.ndim == 1:
126                 return np.concatenate((self.int_2_bin_arr(b),input_data))
127             else:
128                 inp_anz = input_data.shape[0]
129                 b_bin = np.array([self.int_2_bin_arr(b) for ii in
130   range(inp_anz)])
131                 return np.hstack((b_bin,input_data))
```