Machine Learning Based Fault Anticipation for 3D Printing

Dorian Voydie, Louis Goupil, Elodie Chanthery, Louise Travé-Massuyès, Sébastien Delautier

Machine Learning Based Fault Anticipation for 3D Printing ⋆

Dorian Voydie * **Louis Goupil** * **Elodie Chanthery** **
Louise Travé-Massuyès ** **Sébastien Delautier** *

* *Atos, Toulouse, France (e-mail: dorian.voydie@atos.net, lgoupil@laas.fr, sebastien.delautier@atos.net)*
** *LAAS-CNRS, ANITI, Université de Toulouse, CNRS, INSA, Toulouse, France (e-mail: echanthe@laas.fr, louise@laas.fr)*

Abstract: In recent years, 3D printing has seen a stellar rise despite its inability to deliver constant quality goods. This article presents a machine learning experiment that results in a model performing fault prediction, in the sense of forecasting the fault, on the printed parts so that printer parameters can be corrected before the faults appear. This model is able to predict faults in real-time during printing, even in the case of multiple faults. It relies on multiple sensors gathering time-series data during printing, a pre-processing of these data to extract the most relevant features and several machine learning algorithms, each suited and tuned to predict at best each fault. A benchmark for testing and tuning the different algorithms is presented. The resulting model has been implemented on a plastic delta 3D printer and tested for the prediction of eight different faults. The best performing model is a random forest, but decision trees are almost as good while explaining what causes the fault.

Keywords: Machine Learning, Time Series Modeling, Additive Manufacturing, Fault Prediction, Fault Anticipation, Fault Detection

1. INTRODUCTION

Additive Manufacturing (AM) has many advantages over machining, molding and other manufacturing methods (Shahrubudin et al. (2019)). However, defects (called faults in this article) often occur on the parts during the production with no way to guarantee the final quality before printing (Oropallo and Piegl (2016)), which has significant overcost consequences due to material, time and energy losses, particularly for metal AM (du Plessis et al. (2020)).

This article presents an experiment for fault prediction based on machine learning to forecast defects specifically on printed parts (i.e. not on the 3D printer). Predictions are done using pre-processed data including printing parameters as well as data gathered by dedicated sensors.

Explainability of the algorithms is discussed since an algorithm able to predict faults with some level of explainability can give the reasons why the fault will occur and thus indicate corrections to apply before the fault happens.

This research aims at being applied in an industrial context and is backed by field feedback from 3D printing engineers.

A state of the art of machine learning methods applied to 3D printing is assessed in Section 2 and our case study is presented in Section 3. Section 4 presents the machine learning method applied to 3D printing fault prediction. Section 5 concludes the paper and gives some perspectives.

⋆ This research is funded by Atos. This project is related to ANITI through the French "Investing for the Future – PIA3" program under the Grant agreement n°ANR-19-PI3A-0004.

2. STATE OF THE ART

Many machine learning methods have been applied to fault detection or anticipation for 3D printing. Unsupervised methods such as k-means are applied to classify machine states (Uhlmann et al. (2018)). A lot of supervised methods have also been implemented. For instance, deep learning methods such as reservoir computing (Zhang et al. (2021)) or convolutional neural networks trained on images of the printing (Jin et al. (2019)), or even a very simple neural network with only one hidden layer with a detection rate of more than 93% (Yen and Chuang (2022)). Also, non-linear support vector machines (SVM) using the kernel-trick (Delli and Chang (2018)) or least-square loss SVM (He et al. (2018)) have been tried. A mix of an encoder with a generative adversarial network has been successful in classifying printer states (Li et al. (2021)). However, all those methods either cover detection and not anticipation of the fault or do not work in real-time at all (they diagnose the part a posteriori). Our contribution is a predictive identification of faults during printing such as Baumann, Felix and Roller, Dieter (2016) does with image segmentation, but using machine learning and data from sensors.

3. CASE STUDY: ADDITIVE MANUFACTURING WITH A PLASTIC 3D PRINTER

3.1 Principles

Our experiment is led on a microdelta rework printer developed by eMotion Tech. This printer uses plastic

additive manufacturing, meaning it successively deposes thin layers of melted *Poly Lactic Acid* (PLA) at the right coordinates to build the requested part. PLA is a kind of bioplastic (plastic made from renewable, organic sources) that is used in most printers, see Garlotta (2001).

In order to print a part, a 3D model of this part has to be designed. Then, this model is converted into a *Standard Triangle Language* file (STL file for short, originally STereoLithography, see Bártolo (2011)). This file describes only the surface geometry without any representation of color, texture, scale or units. This file is then fed to a *slicer*, a software parameterized with specific printer characteristics, that outputs a file understandable by the 3D printer. In our case, this file is a *gcode* file written in the language Marlin (Krüger et al. (2018)). This file is then given to the printer that prints the part.

It is important to note that the process of slicing is crucial. It is at this stage that most printing parameters are set, and, based on 3D printing experts' experience, it is at this stage that most common printing faults are generated.

The physical model of the 3D printer is not fully known, meaning that we cannot use system equations to model the printer behavior. Hence the use of methods based on data that do not require system knowledge, such as machine learning. This also means that we do not know precisely how the printing parameters influence the behavior of the system. Therefore, when deciding those parameter values for the experiment, we have to be careful not to bias the experiment in a way we think is right despite not knowing the parameters influence.

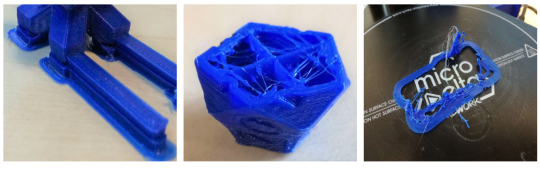
3.2 Fault Types

3D printing having poor performances overall, many people got interested in identifying and classifying its possible faults. For this experiment, it is important to identify the faults we want to be able to predict.

Based on the feedback of experts, eight different types of faults have been selected, divided in 2 categories:

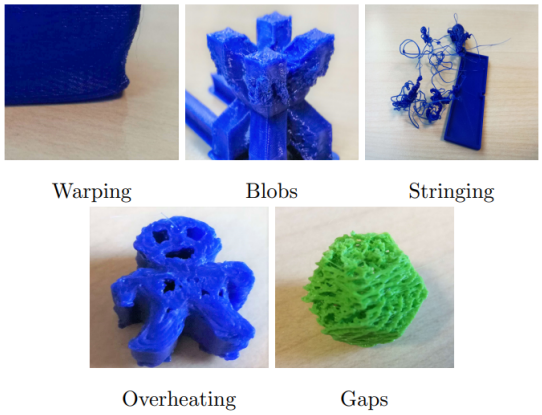
- Critical faults
- Severe faults

When a critical fault occurs, the printed part is expected to be non-functional and so distorted that it becomes useless to continue the printing (see Figure 1). However, when a severe fault occurs, the part remains functional but its aspect integrity is questioned (see Figure 2). Obviously, depending on the aim of the part and the requirements of the part manufacturer, these categories may variate.



Layer Shifting Not Extruding No Adhesion

Fig. 1. Critical Faults



Warping Blobs Stringing

Overheating Gaps

Fig. 2. Severe Faults

4. MACHINE LEARNING FOR 3D PRINTING FAULT PREDICTION

The experiment aims to know whether it is possible to predict faults in real-time during printing with enough accuracy so that the removal of those faults would drastically improve the average quality of 3D printed parts. More specifically, it is to know if this anticipation is possible using machine learning based algorithms trained on data from sensors (that record during printing) and knowledge about the part to be printed in the form of the gcode file. For the purpose of clarity, in this article, the result of the experiment will be called the *software*. This experiment and the resulting software are the main contribution of this article, along with the pre-processing method used.

In order to answer the experiment question and design the software, we equipped the printer with different sensors (see Section 4.1) and gathered data (see Section 4.2). As indicated by Subasi (2020), a lot of pre-processing was required so that the data suitably inputs each algorithm and the performances become acceptable for real-time fault anticipation during printing (see Section 4.3). Then, the data are used to train several machine learning algorithms to detect each previously identified fault (see Section 4.4). Figure 3 provides an overview of the whole process with, in dotted lines, the correction that is not yet implemented but is the end-goal of the project. The results are discussed in Section 4.5.

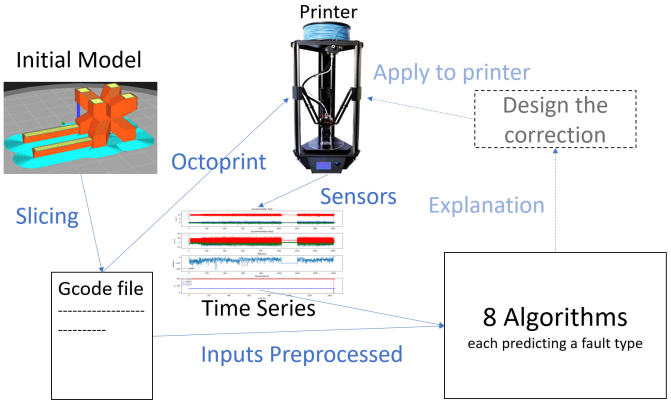


Fig. 3. The Experiment

4.1 Instrumentation

Since we suspect the three factors affecting printing to be initial, environmental and printing parameters, the inputs of the machine learning models are as follows:

- 3D model of the part to be printed in the shape of *gcode*, a code interpretable by the printer that is specific to a printed part and that contains all parameter value instructions for the printing of said part,
- sensor values of printer parameters recorded during printing,
- sensor values of environmental parameters recorded during printing.

We equipped the printer with sensors according to the above items. The printer also has integrated sensors. All the signals retrieved are:

- Nozzle Temperature (actual and target),
- Bed temperature (actual and target)
- Layer information (number, time, mesh, etc.),
- General printing settings of maximum speeds, maximum accelerations, dimension boundaries, printing time, nozzle travel lengths, etc.,
- Nozzle acceleration, gyration and exterior temperature,
- Board acceleration,
- Wire spool weight, equivalent to the wire tension in our case.

4.2 Building a Dataset

Input Data In all machine learning approaches, building a relevant dataset is key. The quality of data is often more important than any kind of tuning done on the hyper-parameters. This is because the key information (the signal signatures that precede a fault) that the algorithm has to predict rightfully needs to be present in the data. However, in our case, as mentioned previously, we do not fully know how the system reacts to some parameters or combinations of parameters, and so we do not know in which signals the fault signatures lie. Thus, we want to get as many signals as input as possible and with the maximum level of accuracy possible. Still, reducing computation time is paramount in real-time problems, hence the absence of camera data that would require heavy image processing.

Labeling Since a goal of the experiment is to know which fault is about to occur, if any, at all times, this means going for a supervised approach. Indeed, an unsupervised approach such as anomaly detection would only predict the occurrence of a fault, but not *which* fault occurs. This leads to having a labeling process. In order to know which fault occurs during each time frame, we recorded it during printing using a synchronized chronometer. This makes 8 time-series (one for each type of fault) of zeros (nominal) and ones (faulty) that last as long as the printing lasts.

An experiment plan was carried out in order to know which parts to print with which parameters for the dataset. It makes sure faults appear with a high enough frequency to get interesting training data without biasing the dataset. Following the experiment plan, the dataset was printed. A total of 54 parts were printed using 6 different shapes and various printing parameters.

In machine learning, once data are obtained it is rarely exploitable right away. Pre-processing is crucial to improve the exploitability of data and the performances of classification, as stated in Subasi (2020).

4.3 Pre-processing data

Retrieving printing data First of all, the signals are synchronised on the highest frequency signal (the board accelerometer frequency). Its period is $3e^{-3}$s. Empty values are filled using linear interpolation (Lepot et al. (2017)).

Once the signals and labels are gathered in the same table, we extract sliding windows of a fixed amount of seconds from the time-series. For this experiment, we chose windows of 10 seconds which corresponds to 3330 time-steps, with a stride half the size of the window, meaning two successive windows share half of their data.

We end up with a 3-dimensional table with dimensions:

$$(n_{windows}, 3330_{time-steps}, X_{variables} + Y_{labels})$$

where $n_{windows} = \frac{PrintingTime(s)}{10}$ for a printing.

The same process with windows of 1 second and windows of 50 seconds was implemented. Multiple sizes are tested because, again, the behavior of the fault signatures is not accurately known. With a stride of half the window size, a 1 second window allows the algorithm to compute for 0.5 second. Less than that would be too much of a constraint for a machine learning algorithm. Meanwhile, a 50 seconds window should identify most signatures (once again, according to 3D printing experts). The 10 seconds window is a good trade-off.

Once the windows are extracted, they must be labeled. It was decided to label every window as faulty (1) or (normal (0) according to each type of fault. Each algorithm gets fed the whole window and is trained to output 0 or 1 for its corresponding faults. For critical faults, the presence of a faulty time-step in the window (even only one) results in the window being labeled as faulty. Meanwhile, for severe faults, if the majority of the time-steps are faulty the window is labeled as faulty for this fault, otherwise it is nominal.

Let us note that this labeling allows detection of faults but not anticipation. To be able to train the algorithms to *predict* faults, the label is shifted one window backwards. It means that a window is labeled with a fault that will occur during the next window.

In order to make data intelligible to algorithms, feature engineering is performed on these data. The list of features extracted from every window is shown in Table 1. To enhance algorithm performances and inference time, feature selection is also performed before training each algorithm using scikit-learn[1].

The windows, now characterized by the features, are split 80%/20% between a train-set and a test-set. Because algorithm performances can greatly vary according to the data put in each set, two means of splitting the dataset are used:

Table 1. Features

Name	Signification
mean	Mean value
std	Standard deviation
mad	Median absolute deviation
max	Largest value in array
min	Smallest value in array
iqr	Interquartile range
maxPeak	Largest frequency value
meanFreq	Frequency signal weighted average

- **Statistical split**: separate the windows between training and testing randomly and regardless of which printing they belong to.
- **Objective split**: separate the printings between training and testing and then extract the windows. Two windows from the same printing can not be in different sets.

Once the splits are done, within each set and for each fault type separately, the classes are balanced so the algorithms do not get biased. More often than not, there are more nominal than faulty windows so this means removing some nominal windows. We used a *One-versus-Rest* (See Tax and Duin (2002)) method to create our balanced datasets. This means that for each fault we have a dataset with half the windows containing said fault, and the other half a random set of windows with any combination of nominal or the other faults.

4.4 Training and results

In order to obtain relevant results, a 10-*Folds Cross Validation* (Refaeilzadeh et al. (2009)) is applied on the training sets to assess how the results of the training generalize to a new set of printings.

For each of the eight fault types, six algorithms were trained (See Singh et al. (2016) and Izenman (2013)). These algorithms and their main hyper-parameters are presented in Table 2.

Table 2. Algorithms and Hyper-parameters

Algorithms	Acronyms	Hyper Parameters	Values
Logistic Regression	LR	C solver multiclass penalty	$\in]0, 5]$ "liblinear" "auto" "l1"
Linear Discriminant Analysis	LDA	$n_{components}$	100
K-Nearest Neighbors	KNN	$n_{neighbors}$	3
Random Forest	RF	$n_{estimators}$	400
Decision Tree	DT	max_depth	None
Multi-layer Perceptron	MLP	hdn_layer_sizes activation	(64,32,16,) "tanh"

The algorithms presented in Table 2 are all implemented using the scikit-learn library for Python. The hyper-parameters descriptions can be found in the scikit-learn documentation [1] . The tuning of these hyper-parameters is mostly done by performing a grid search in commonly used intervals.

The performances of these algorithms on windows of 1 second are shown in Table 3 and Table 4 with *Mean* being the average accuracy on each validation of the cross-validation on the testing set and *STD* the standard deviation. In the statistical split, the RF algorithm is outperforming other models by a large margin, and as expected, models trained on the statistical split are more often than not better than the ones trained on the objective split.

Decision Trees On Figure 4, a DT trained to predict the fault "No Adhesion" is presented. The first criterion to discriminate between faulty and nominal is whether the bed temperature is above or below 31°C. Indeed, with a bed (or board) temperature too low, the printed part does not stick to the bed. DTs are not only one of the best performing algorithms but also present a level of explainability that no other algorithm tested here do.

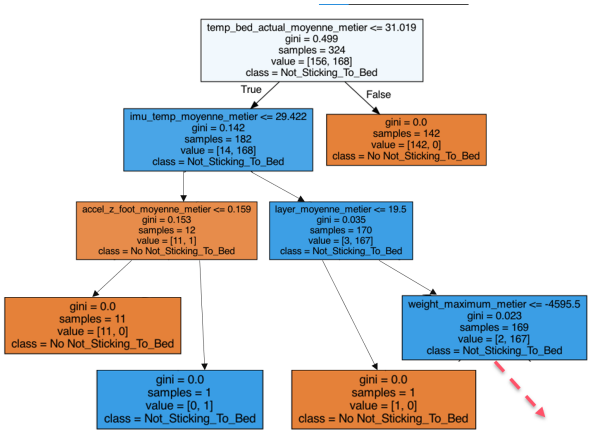


Fig. 4. Decision Tree Trained to Predict the No Adhesion Fault (Blue=Faulty, Orange=Nominal)

Random Forest RF is an aggregate of DTs. Multiple DTs are built during the training phase and the final prediction is obtained by mixing the results together. RF is the only algorithm that performs better than DTs and hence it outperforms all the tested algorithms. However, it looses the explainability power of DTs.

The consequence of the obtained results is that the present version of the software used to predict faults in real-time on the printer at Atos is based on DTs used to detect each type of fault. These indeed achieve a good tradeoff between performance and explainablility.

4.5 Discussion and perspectives

In this section we analyze the results and what should be done to improve the method.

Split and Increase the Dataset In Section 4.3 we mentioned the 2 manners to split the dataset. The objective split is relevant to our application. Indeed, we want the algorithms to be able to predict faults on new printings that do not already belong to the dataset. However, the performances in the objective split are lackluster because some faults are not represented in the training dataset.

[1] https://scikit-learn.org/stable/modules/classes.html

Table 3. Statistical Split

	LR		LDA		RF		DT		MLP		KNN	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Warping	88.47	1.01	93.3	0.61	99.85	0.12	98.91	0.36	67.65	11.85	88.4	0.82
Layer Shifting	69.85	8.22	67.43	6.54	90.99	4.67	84.46	4.99	69.44	6.36	72.99	7
Blobs	72.67	0.66	93.14	0.28	99.79	0.08	99.19	0.19	53.75	5.75	80.49	0.54
No Adhesion	85.40	2.30	96.41	1.35	99.46	0.42	98.15	0.50	73.76	8.91	81.21	2.08
Stringing	73.6	0.31	95.29	0.26	99.94	0.04	99.84	0.04	52.81	1.92	84.13	0.33
Gaps	77.67	0.46	96.53	0.22	99.71	0.09	99.67	0.14	53.56	3.11	78.83	0.81
Overheating	60.6	0.69	91.94	0.28	99.94	0.03	99.73	0.06	53.47	1.23	81.28	0.31
Not Extruding	70.09	2.16	92.31	0.7	99.22	0.35	97.59	0.44	65.51	8.74	84.93	0.83

Table 4. Objective Split

	LR		LDA		RF		DT		MLP		KNN	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Warping	67.24	13.83	51.43	17.67	59.10	13.20	57.84	15.70	56.78	11.95	53.35	11.31
Layer Shifting	50.4	25.57	53.67	24.11	48.03	11.22	54.72	20.80	49.41	19.97	47.11	18.36
Blobs	61.49	8.96	40.82	9.59	49.25	3.48	47.15	6.43	52.74	4.21	48.39	7.71
No Adhesion	55.77	16.98	46.49	5.73	50.49	2.48	52.51	6.9	56.92	15.58	53.41	13.36
Stringing	66.42	16.93	59.60	15.35	65.64	16.59	63.38	16.11	42.66	10.36	53.61	9.41
Gaps	61.10	17.23	59.24	21.14	54.23	5.11	56.69	9.83	42.57	9.43	47.75	9.4
Overheating	47.85	16.59	57.24	15.85	58.7	14.32	59.69	14.85	45.45	11.35	46.64	7.79
Not Extruding	59.29	14.2	65.75	15.78	73.58	22.97	68.02	18.70	53.40	15.56	56.38	12.01

Indeed, some faults appear in only a few printings, meaning that if those printings are all in the testing set, the algorithms are not trained on them and thus not able to predict them. And vice versa. As a result, despite the objective split being more suitable for the goal, its performances are nerfed by the way the dataset is built. Hence the statistical split.

After these promising results of the presented experiment, the dataset extension is still ongoing, hence the low number of fault occurrences. The aim is, for a big enough dataset, to make both splitting approaches equivalent. Indeed, if more parts are printed and included in the dataset, the case where a fault type is not present in either the training set or the testing set is less likely to happen. Also, the main consequence of choosing one splitting method over the other is the different degree of similarity between the training and testing sets. With a large enough dataset, this difference disappears and the performances should also become equivalent. This is why using the statistical split makes sense. Still, for now we can not prove this point so the results from the objective approach should be taken with a grain of salt. In a more generic sense, improving the dataset should definitely improve the performances. In particular, the focus has to be on increasing the presence of rare faults and increase the number of printings they appear in.

Improve the performances For now, the good prediction rate of the software is not high enough to drastically improve the quality of a new printed part. It cannot be used in an industrial environment in its current state because of the accuracy reached in the objective split (that is equivalent to testing the software on a new, unknown printed part). Indeed, using the objective split and taking the best algorithm for each fault type, several faults are still predicted correctly in only 55% of the cases, which is just slightly better than a coin flip.

As a last resort, if extending the dataset fails to provide valuable results, we can assume that the information we want the algorithm to learn from is not present in the data.

Geometrical Analysis The 3D printing experts all agree on the fact that the geometrical shape of the part to be printed is a very important factor for faults. In this experiment, the dataset was built using 6 different shapes for its parts. A geometrical shape is a combination of many basic shapes (such as pyramids, cylinders and whatnot). Perhaps, a geometrical study could help prove that all 3D parts are made of a finite set of elementary shapes and this elementary shapes could be used to train the fault prediction algorithm, thus increasing its robustness to new, unknown shapes.

Computer Vision In order to identify anomalies the printer has been equipped with two cameras. There is a possibility that these cameras can detect precursors to specific types of fault. The reason we think it is possible is because it actually happens on the printer we use at Atos. At the beginning of each printing, when the nozzle heats up before the printing, a small blob of PLA forms at the tip of the nozzle. When the printing starts it can remain embedded in the part and cause a blob on it. Some works such as Delli and Chang (2018) are able to detect faults when they occur, in this case using *Support Vector Machines*. However, once the fault happens, it is too late. Henceforth, we have to detect precursors. Since we do not know what form those precursors could take, it would require either to compare the current printed part with its 3D model (may be with a geometrical analysis similar to what the authors in Petsiuk and Pearce (2020) did) or to train a computer vision algorithm to predict the fault using labels from our already built dataset.

Hybrid Methods It may be that, even if the amount of data, its precision, and its quality are increased, the algorithms are not able to predict the faults. A possible reason for that would be because the algorithm structures

are not able to represent the real system behavior. This goes back all the way to the fact that the mathematical equations that govern the behavior of the system are not fully known. This prevents the use of a model-based method. However, nowadays, many hybrid methods for diagnosis such as Yang (2004) have been developed to compensate for the lack of knowledge about the system with data. A starting point could be to estimate the system equations using machine learning (for instance with symbolic regression, see Quade et al. (2016)). An advantage of most hybrid methods over pure data-based methods is explainability of the outcome. In order to eventually correct the faults in real-time, explainability is a must-have.

5. CONCLUSION

In this paper, we describe the experiment we led on predicting faults on 3D printed parts. The experiment aims at designing a software that anticipates eight different faults in real-time using data from sensors monitoring the printer and six different machine learning algorithms each trained for each fault type. The results show that the Random Forest algorithm has the best results to predict all types of faults. However, the performances for new, unknown, printed parts leave a lot to be desired. Multiple suggestions are made to improve the algorithm accuracies. Expanding the dataset, performing a geometrical analysis of the part, adding data from cameras enhanced with computer vision algorithms, or combining a model-based method with the machine learning approach.

If a very high precision on the software prediction is reached, the next step is to explain its prediction. This means finding out the root cause of the failure in order to perform a real time correction of the fault. As mentioned previously, this is something to keep in mind when choosing which algorithms to use, or how to design the method.

Then, the final goal is to be able to translate everything that works on the printer on which the method is tested, to other, different, types of printers.

REFERENCES

Bártolo, P.J. (2011). *Stereolithography: materials, processes and applications*. Springer Science & Business Media.

Baumann, Felix and Roller, Dieter (2016). Vision based error detection for 3d printing processes. *MATEC Web of Conferences*, 59, 06003.

Delli, U. and Chang, S. (2018). Automated process monitoring in 3d printing using supervised machine learning. *Procedia Manufacturing*, 26, 865–870.

du Plessis, A., Yadroitsava, I., and Yadroitsev, I. (2020). Effects of defects on mechanical properties in metal additive manufacturing: A review focusing on x-ray tomography insights. *Materials & Design*, 187, 108385.

Garlotta, D. (2001). A literature review of poly(lactic acid). *Journal of Polymers and the Environment*, 9(2), 63–84.

He, K., Yang, Z., Bai, Y., Long, J., and Li, C. (2018). Intelligent fault diagnosis of delta 3d printers using attitude sensors based on support vector machines. *Sensors*, 18(4).

Izenman, A.J. (2013). Linear discriminant analysis. In *Modern multivariate statistical techniques*, 237–280. Springer.

Jin, Z., Zhang, Z., and Gu, G.X. (2019). Autonomous in-situ correction of fused deposition modeling printers using computer vision and deep learning. *Manufacturing Letters*, 22, 11–15.

Krüger, J., Gu, W., Shen, H., Mukelabai, M., Hebig, R., and Berger, T. (2018). Towards a better understanding of software features and their characteristics: A case study of marlin. In *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems*, 105–112. Association for Computing Machinery.

Lepot, M., Aubin, J.B., and Clemens, F.H. (2017). Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water*, 9(10).

Li, C., Cabrera, D., Sancho, F., Sánchez, R.V., Cerrada, M., and de Oliveira, J.V. (2021). One-shot fault diagnosis of three-dimensional printers through improved feature space learning. *IEEE Transactions on Industrial Electronics*, 68(9), 8768–8776.

Oropallo, W. and Piegl, L.A. (2016). Ten challenges in 3d printing. *Engineering with Computers*, 32(1), 135–148.

Petsiuk, A.L. and Pearce, J.M. (2020). Open source computer vision-based layer-wise 3d printing analysis. *Additive Manufacturing*, 36, 101473.

Quade, M., Abel, M., Shafi, K., Niven, R.K., and Noack, B.R. (2016). Prediction of dynamical systems by symbolic regression. *Physical Review E*, 94(1).

Refaeilzadeh, P., Tang, L., and Liu, H. (2009). *Cross-Validation*, 532–538. Springer US.

Shahrubudin, N., Lee, T., and Ramlan, R. (2019). An overview on 3d printing technology: Technological, materials, and applications. *Procedia Manufacturing*, 35, 1286–1296.

Singh, A., Thakur, N., and Sharma, A. (2016). A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 1310–1315.

Subasi, A. (2020). Chapter 2 - data preprocessing. In A. Subasi (ed.), *Practical Machine Learning for Data Analysis Using Python*, 27–89. Academic Press.

Tax, D. and Duin, R. (2002). Using two-class classifiers for multiclass classification. In *2002 International Conference on Pattern Recognition*, volume 2, 124–127 vol.2.

Uhlmann, E., Pontes, R.P., Geisert, C., and Hohwieler, E. (2018). Cluster identification of sensor data for predictive maintenance in a selective laser melting machine tool. *Procedia Manufacturing*, 24, 60–65.

Yang, Q. (2004). *Model-based and data driven fault diagnosis methods with applications to process monitoring*. Case Western Reserve University.

Yen, C.T. and Chuang, P.C. (2022). Application of a neural network integrated with the internet of things sensing technology for 3d printer fault diagnosis. *Microsystem Technologies*, 28(1), 13–23.

Zhang, S., Duan, X., Li, C., and Liang, M. (2021). Pre-classified reservoir computing for the fault diagnosis of 3d printers. *Mechanical Systems and Signal Processing*, 146, 106961.