Detecting voids in 3D printing using melt pool time series data

Vivek Mahato[1] · Muhannad Ahmed Obeidi[2] · Dermot Brabazon[2] · Pádraig Cunningham[1]

Abstract

Powder Bed Fusion (PBF) has emerged as an important process in the additive manufacture of metals. However, PBF is sensitive to process parameters and careful management is required to ensure the high quality of parts produced. In PBF, a laser or electron beam is used to fuse powder to the part. It is recognised that the temperature of the melt pool is an important signal representing the health of the process. In this paper, Machine Learning (ML) methods on time-series data are used to monitor melt pool temperature to detect anomalies. In line with other ML research on time-series classification, Dynamic Time Warping and k-Nearest Neighbour classifiers are used. The presented process is effective in detecting voids in PBF. A strategy is then proposed to speed up classification time, an important consideration given the volume of data involved.

Keywords Process monitoring · Classification · Time-series

Introduction

This study is focused on metal powder printing, an additive manufacturing process that is having a high impact in industries such as automotive, aerospace and biomedical. In additive manufacturing, functioning parts can be produced directly from a computer aided design model without the need for moulding or prototyping. The PBF process is shown in Fig. 1. The part is built as the laser traces out the part in layers in the powder bed.

The overall process is complex and build quality is very sensitive to process parameters (Aminzadeh and Kurfess 2019; Kwon et al. 2020). For this reason, there has been considerable research on analytics methods for in-situ monitoring to detect the onset of defects. Grasso and Colosimo (2017) provide a comprehensive review of this research. It is clear that the temperature of the melt pool is an important signal for monitoring build quality. In this paper we explore the hypothesis that anomalies such as pores can be detected by identifying characteristic signatures in melt pool time-series data.

While the ultimate objective is to detect pores arising from incorrect process parameter, the data analysed here comes from blocks containing voids by design. The size of these voids is set to correspond to the size of pore defects arising during normal PFB machine part production. The fact that the void locations are known by design provides a ground-truth to assess the performance of the ML methods.

Our analysis uses data from the Aconity MINI 3D printer.[1] The temperature data comes from pyrometers that monitor melt pool temperature. Two pyrometers supplied by KLEIBER Infrared GmbH detect the heat emission light in the range of 1500 to 1700nm via two detectors. The pyrometers track the scan of the laser to provide a time-series (see Fig. 2) sampled at 100 Hz. Based on our core hypothesis that the presence of pores will have characteristic signatures in the pyrometer time-series data, the pore detection task is cast as a time-series classification problem. The void shown in Fig. 2a shows up as a dip in the time-series in Fig. 2b and c. So the signatures are dips of varying size in the time-series. Other types of defect could show up as spikes in the time-series, e.g. when the void results from a rise in melt-pool temperature due to laser beam power instability issues. The same time-series classification strategy will apply.

Many ML methods are not applicable for time-series classification because they require a feature vector representation. However, variable length time-series data can by handled by k-Nearest Neighbour (k-NN) classifiers if effec-

✉ Pádraig Cunningham
   padraig.cunningham@ucd.ie

   Dermot Brabazon
   dermot.brabazon@dcu.ie

[1] University College Dublin, Dublin, Ireland
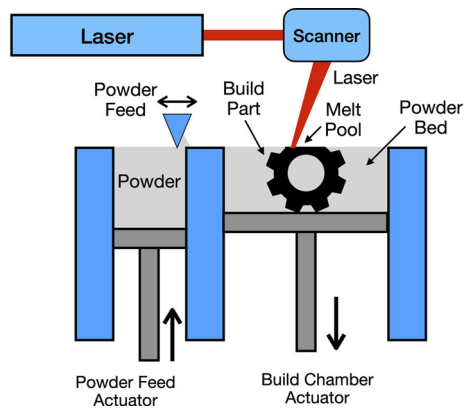
[2] Dublin City University, Dublin, Ireland

---

[1] https://aconity3d.com/products/aconitymini/

Fig. 1 The powder bed fusion process

Fig. 2 Representation of the data collected during additive-manufacturing of a stainless steel block. The heatmaps show the voltage signal from the pyrometer which is directly proportionate to the temperature

tive distance measures can be identified. We evaluate three candidate distance measures (see "Machine learning on time-series data" section) and find that Dynamic Time Warping (DTW) is effective for this PBF data. A standard print run produces a vast volume of data and DTW is computationally expensive so we evaluate a number of strategies to speed up the classification process.

The PBF process and the data we use for our analysis is described in more detail in the next section. In "Machine learning on time-series data" section we review the relevant ML methods and in "Evaluation" section we present our evaluation.

3D Printing using PBF

In PBF, parts are designed by using CAD software, the CAD model is then saved in a file format which the metal 3D printer will accept. The model is then sliced into two dimensional layers which the fusion source (e.g. laser beam) traces out during part production. The total number of layers is equal to the maximum height of the part divided by the layer thickness. The laser melts the metal powder particles at the positions where the 2D plane intersects with the part, hence the term selective laser melting (SLM) which is also used to describe the process. Once the fusion of the layer is completed, the build plate moves down a distance equal to the layer thickness and a new powder layer is spread onto the previously sintered 2D layer (see Fig. 1). During the melting of the new layer, part of the previously solidified layer is re-melted to enhance the chemical bonding in a process similar to the micro-welding. The melting and deposition of the metal powder layers continues through the full set of layers in the slice file until the 3D part is completed.

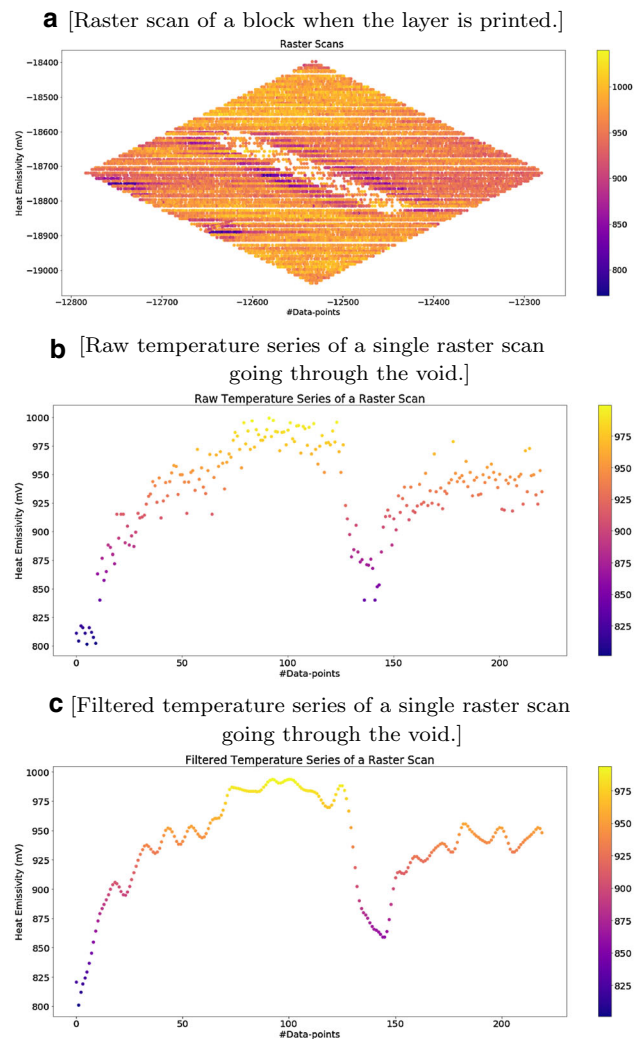An important advantages of the AM manufacturing method over conventional methods is the material saving. In AM, only the exact amount of metal powder required in the part volume will be used. The un-fused powder can be recycled and used in the manufacture of other parts. Also, parts made of several components can be designed and manufactured in a single build process and thus overcome the need for assembly which is required when using the conventional methods (Fitzsimons et al. 2019). On the other hand, due to the long build time, additive manufacturing is not typically economical when mass scale production is required compared to other methods like die casting and metal injection moulding. In addition AM requires a significant up-front investment in machinery and some specialised powders are expensive compared to conventional feedstock (Bose et al. 2018; Demir et al. 2017).

Issues with PBF

Despite the progress in metal AM technology, the challenges for achieving optimal process control are still considerable. There are still difficulties in achieving process repeatability. Difficulties can relate to porosity, grain and microstructure defects, and with grain orientation and compositional control (Bhavar et al. 2017; Gibson et al. 2014; Mani et al. 2017). In PBF, thermal gradients are controlled via liquid, semi-solid and gaseous convection, conduction through the part and powder, and the input heat source fluence and scan strategy.

Variations in part structure and properties are seen in parts using the same make and model of metal AM machines even when the same processing parameters are applied. Other factors that lead to reduced process repeatability include variability in the metal powder supply, the build chamber volume and the arrangement of the parts within the build chamber, the inert gas flow design and flow rate, oxygen gas content and the build chamber global temperature. Given these complexities, in order to control the process, in-situ monitoring of the build process is required. Data can be gathered and analysed in real time in order to anticipate defects. The response may be to generate an alarm or even alter process parameters to avoid defects.

For our analysis we use a similar methodology to that used by Grasso et al. (2018). They used a thermal infrared camera to capture images and measure data from the build process of 5 mm zinc cubes at a frequency of 50 Hz. The aim of their study was to anticipate out-of-control conditions.

PBF monitoring using machine learning

Grasso and Colosimo (2017) provide a review of commercial tools for PBF process monitoring. State-of-the-art systems provide for in-situ sensing and some post-process data reporting. In principle, process parameter adjustments could be made between layers if the data analytics could be completed on time (Mani et al. 2017).

Figure 2 helps illustrate what this would entail in our framework. The data to be analysed for each layer is the pyrometer data for that layer (Fig. 2a). Our strategy is to divide the layer data into individual scans and classify individual scans as normal or anomalous. The parts analysed have approximately 85 lines/scans in each layer. Really short scans are excluded from consideration leaving on average 57 scans to be analysed for each layer. Each of these scans is represented as a time-series and these time-series can be classified as anomalous or normal using the ML methods described in "Machine learning on time-series data" section.

Applying the algorithm speedup described in "k-NN speedup" section , a time-series for a scan takes 0.8ms to classify. So checking a single layer takes approximately 45ms. It

Table 1 Summary of the data per group

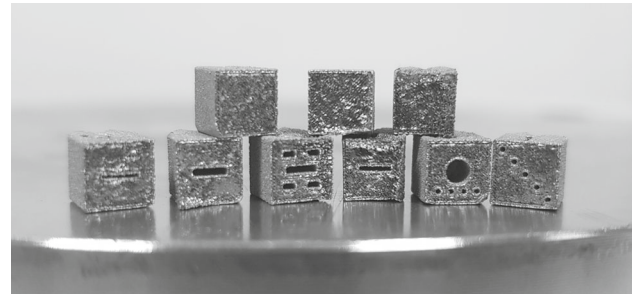| Name	Pore size (mm)	#Pore	#No pore
P_{large}	0.4, 0.5, 0.6	1221	2366
P_{small}	0.1, 0.05	249	632

Fig. 3 Picture of 5 mm stainless steel (316L) cube variants produced using the Aconity MINI 3D printer with pyrometer data for analysis

takes 10 to 20 seconds to print a layer so the ML assessment is much faster than the printing process. This would allow for an intermittent controller method of processing, whereby after each layer is processed, the areas where void defects were detected would then be retraced in order to remelt the surface and close off the pore.

Detecting pores

For our first analysis we consider the six 5 mm cubes in the P_{large} dataset—see Table 1 and Fig. 3. These contain voids of size 0.4, 0.5 and 0.6 mm as can be seen in the figure. The objective is to detect the voids in the pyrometer data—sample pyrometer signals are shown in Fig. 2. The pyrometer sampling frequency is 100 kHz. This produces a melt pool temperature measurement every $10\mu s$; e.g. one measurement in each one micron in the x and y directions based on a scanning speed of 1000 mm/s. For the purpose of classification, time-series samples are extracted as explained in "PBF monitoring using machine learning" section. The P_{large} dataset contains 1221 samples with pores and 2366 samples without pores.

The data is organised into two datasets as summarised in Table 1. The P_{large} dataset covers large pores (0.4–0.6 mm). For these cubes, the laser is set at 180 W, the scanning speed was 800 mm/s and the layer thickness was $20\mu m$. The The P_{small} dataset covers small pores (0.05–0.1 mm) with the same process parameters except the thickness is set at $40\mu m$.

It is clear from the graph in Fig. 2 b that the pyrometer data was quite noisy. For this reason we considered two versions of the data, the raw data and filtered data passed through a low-pass Butterworth filter (sample rate = 100 Hz, cutoff = 10 Hz and order = 4).

Machine learning on time-series data

Time-series data presents a significant challenge for ML because most of the state-of-the-art ML methods require data in a feature vector format. This is true for both supervised (Bagnall et al. 2017) and unsupervised (Aghabozorgi et al. 2015; Wu et al. 2005) ML research on time-series data. In this research we have access to labeled data so we are frame the task as a supervised ML problem.

Variable length time-series can be 'normalized' into a feature vector but, as we will see in Fig. 8, results using Euclidean distance on normalised time-series data are not very good. However variants on nearest neighbour methods (k-NN) and kernel methods are applicable if appropriate distance metrics or kernel functions can be identified (Bagnall et al. 2017).

In this study we focus on k-NN and on effective distance measures for time-series. Because these distance measures are computationally expensive we also consider methods for speeding up k-NN.

Distance measures for time-series

Our evaluation covers three distance measures that have been developed for time-series data. These are Dynamic Time Warping, Symbolic Aggregate approXimation and Symbolic Fourier Approximation. These are each described here; before that we discuss how Euclidean distance can be made work for time-series data.

Making Euclidean distance work

Euclidean distance is probably the most popular distance metric in ML however, as stated already, it is not directly applicable with time-series data. It is not suitable because it forces an alignment of the time-series points. It will not even be applicable if the time-series are of different lengths. Multiple techniques are present in the literature to handle this variability in the series data, a few of which are as follows:

1. *Padding* Padding is the most popular choice in the literature, it involves extending short series with fixed value entries. The padding is done either at the start (prepadding), or the end (post-padding) or at both ends. It is normal to pad with zeros but the mean or the median of the series can be used as well.
2. *Truncation* An alternative strategy is to truncate longer series. Again, the truncation can be done from the start (pre-truncation), the end (post-truncation), or both.
3. *Interpolation* This strategy indicates interpolating all the series to the length of the longest series present in the dataset. Interpolation is achieved by using curve-fitting functions, such as splines or least-squares, to name a few.
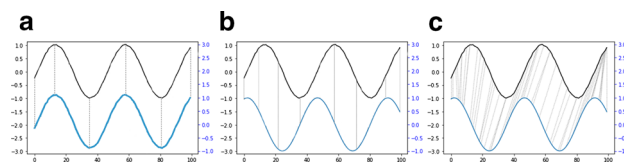

Fig. 4 Dynamic Time Warping; **a** Two similar time-series— Euclidean distance is small. **b** Two similar time-series displaced in time— Euclidean distance is large. **c** An example DTW mapping for the two time-series in **b**

In this study, we employ a combination of Truncation and Padding techniques to allow Euclidean distance to be used. The canonical length is set to 259 and series are post-truncated or post-padded with zeros as appropriate. Series of length less than 100 are removed.

Dynamic time warping

The problem with using Euclidean distance on time-series data is illustrated in Fig. 4. Euclidean distance is effective if two time-series are very well aligned (see Fig. 4a). However, if the two time-series has even a small misalignment, it can result in a sizeable Euclidean distance (Fig. 4b). DTW attempts to address this misalignment by providing us with flexibility of mapping the data-points in a non-linear fashion (Fig. 4c) (Mahato et al. 2020).

The DTW computes the distance between two time-series **q** and **x** as follows:

$$DTW(\mathbf{q}, \mathbf{x}) = \min_{\pi} \sqrt{\sum_{(i,j) \in \pi} d(q_i, x_j)^2} \qquad (1)$$

where $\pi = [\pi_1, \ldots, \pi_l, \ldots, \pi_L]$ is the optimum path (mapping) having the following properties:

- $t_1 = |\mathbf{q}|, t_2 = |\mathbf{x}|$
- $\pi_1 = (1, 1), \pi_L = (t_1, t_2)$
- $\pi_{l+1} - \pi_l \in \{(1, 0), (0, 1), (1, 1)\}$

DTW constructs a cost matrix where it populates each cell (i, j) with the distance between q_i and x_j. The algorithm then finds the shortest path through the grid. The cumulative distance along this path is the overall distance between the two time-series. The magnitude of the deviation from the actual diagonal of the matrix reflects the *warping*. The computational complexity of the DTW algorithm is $O(t_1, t_2)$ as it involves a search through the matrix. This complexity is effectively $O(t^2)$ in the average length of the time-series— so DTW is computationally expensive. The performance of DTW is enhanced by employing the Sakoe and Chiba (1978) global constraint in the model. This reduces the time and memory complexity of the algorithm.
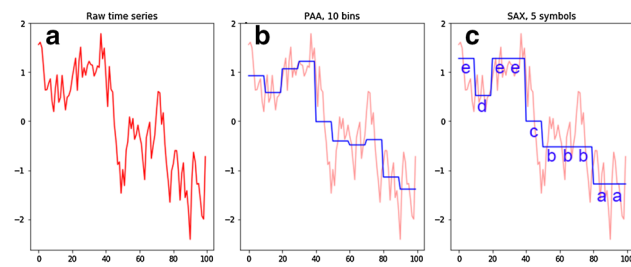
Fig. 5 Symbolic Aggregate Approximation; The raw time-series in **a** will be represented by the sequence edeecbbbaa in **c**

Symbolic aggregate approximation

There has been considerable research in the past decade on developing symbolic representations for time-series data. The main motivation is to use the potential of already established text processing algorithms to solve time-series tasks. A summary of such methods is provided by Lin et al. (2003).

Symbolic Aggregate Approximation (SAX) is one such algorithm which works by converting a time-series into a sequence of symbols. It also tries to obtain dimensionality and numerosity reduction (i.e. more compact representation) of the original time-series. Such transformations present a distance measure which is lower bounding on corresponding measures on the original series (Lin et al. 2003).

SAX employs the Piecewise Aggregate Approximation (PAA) discretisation technique for dimensionality reduction. Due to the difficulty in comparing two time-series of different scales, SAX normalises the original series so that the mean is zero and the standard deviation is one, before passing to PAA for transformation (Lin et al. 2003; Keogh and Kasetty 2002). SAX passes the PAA transformed series through another discretisation procedure that converts them into symbols.

Following the transformation of all the time-series data in our dataset into their symbolic representation, we can readily compute the distance between two time-series by using any string metric. One such popular string metric is Levenshtein distance (Yujian and Bo 2007).

More details of the algorithm are provided in the papers by Lin et al. (2003) and Mahato et al. (2020).

Symbolic Fourier approximation

Symbolic Fourier Approximation (SFA) (Schäfer and Högqvist 2012) is an example of another algorithm like SAX, built on the idea of dimensionality reduction by symbolic transformation. SAX seeks to retain the data in the time-domain, whereas SFA converts the data to bring it in the frequency domain. As in the frequency domain, each dimension then has relative knowledge of the complete series.

SFA, unlike SAX, employs Discrete Fourier Transformation (DFT) as its dimensionality reduction technique to
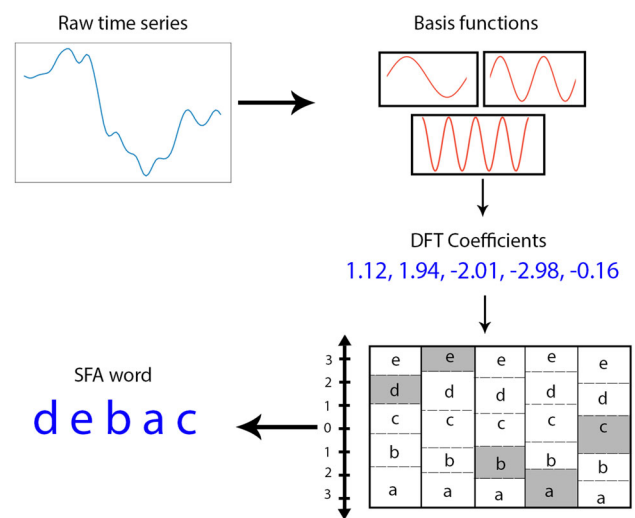

Fig. 6 Symbolic Fourier Approximation; The raw time-series will be represented by the sequence debac Mahato et al. (2019)

focalise the data in the frequency domain. The SFA algorithm applies DFT approximation as part of preprocessing by replacing the original time-series into a series of DFT coefficients. SFA then employs Multiple Coefficient Binning (MCB) process to compute multiple discretisations of the coefficients series (see Fig. 6).

More details of the algorithm are provided in the papers by Mahato et al. (2020) and Schäfer and Högqvist (2012).

k-NN speedup

The big issue with these methods is that they are computationally expensive and this coupled with the large datasets that arise in AM results in classification times that may be prohibitive. For instance DTW has time-complexity $O(t^2)$ where t is the average time-series length. So k-NN classification is $O(nt^2)$ where n is the size of the training set.

Numerous computational tricks exist to speed up nearest neighbour retrieval but these typically depend on one of two things:

– The data has a vector (feature) space representation. Kd-Trees can reduce k-NN retrieval time to $O(\log(n))$ for vector space data in some circumstances. Random projection trees can be used to implement approximate k-NN (Bernhardsson 2016) with significant speed-up but will only work for vector space data.
– The distance measure is a metric. For instance, the use of Ball Trees will speed up k-NN retrieval but the distance measure must be a metric.

Unfortunately neither of these apply for our time-series distance measures because they are not proper metrics and as

explained in "Making Euclidean distance work" section a time-series is not a vector space representation. Although by normalising time-series to the same length and using Euclidean distance, we are effectively treating the time points as features in a vector space. In our evaluation ("Evaluation" section) we test a two-stage strategy where we use approximate Euclidean distance retrieval to return a large set (400) of candidate neighbours and then use DTW to select nearest neighbours among these.

The other strategy for speeding up k-NN is to reduce n by removing redundant training data. In our evaluation we consider two instance selection strategies, Condensed Nearest Neighbour (CNN) and Conservative Redundancy Removal (CRR) (Cunningham and Delany 2020).

Approximate nearest neighbour

Similarity-Based retrieval is of fundamental importance in many internet applications. Because of the size of the data resources involved the linear time requirement of exact retrieval is not realistic so approximate methods have received a lot of attention. However, it is in the nature of typical internet applications that approximate retrieval is often adequate. A number of innovative solutions for approximate nearest neighbour (ANN) retrieval have emerged and some companies have made these generally available (Aumüller et al. 2017).

Researchers at Facebook have released FAISS (Facebook AI Similarity Search) a library for fast ANN into the public domain (Johnson et al. 2017). It uses product quantization (PQ) codes to produce an efficient two-level indexing structure to facilitate fast approximate retrieval. Spotify have shared an ANN implementation called Annoy that uses Multiple Random Projection Trees (MRPT)(Bernhardsson 2016). While tree structures such as Kd-Trees can be used to speed up exact NN retrieval the improvement may be negligible for high dimension data (Cunningham and Delany 2020). This is because the benefits of the tree as an indexing structure are lost due to extensive backtracking. The solution with MRPT is not to perform any backtracking but build multiple trees.

The search for the nearest neighbour is done by locating the query to leaf nodes in all the trees. The neighbourhood is the union of the training points at these leaf nodes. In the example in Fig. 7 this neighbourhood is $\{B, C\} \cup \{A, C, D\}$. Annoy then sorts these candidate points by their distances against query point, and returns top k neighbours.

Evaluation

The objective of our evaluation is to identify the best overall ML strategy for identifying pores of the type described in
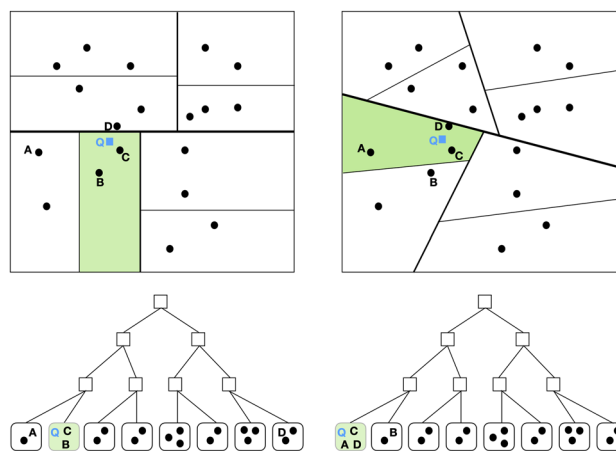

Fig. 7 Two random projection trees in a 2D space. The query Q gets indexed to different regions in the different projections, with B,C on the left and with A,D,C on the right
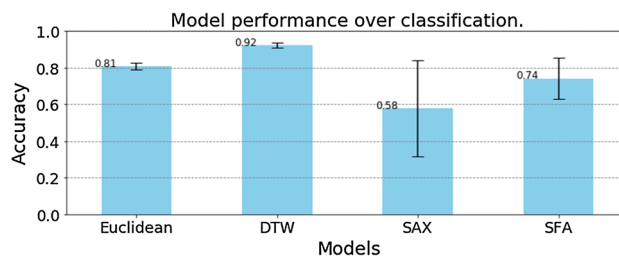

Fig. 8 The performance of models through cross-validation over the filtered temperature-series data

"Detecting pores" section. we consider the following questions:

- What is the best distance measure for this data?
- Is it worth cleaning (filtering) the signal?
- What algorithm speedup strategies are effective without loss of accuracy?
- What is the impact of pore size on classification accuracy.

These questions are addressed in turn in the following sections. Unless otherwise indicated the evaluation is performed using 10-fold cross-validation on the P_{large} dataset described in "Detecting pores" section.

Alternative distance measures

First we assess the effectiveness of the three distance measures described in "Machine learning on time-series data" section. We also consider Euclidean distance as a baseline. In this evaluation the data has been smoothed using a Butterworth filter as described in "Detecting pores" section. The results of a 10-fold cross-validation on the P_{large} data are shown in Fig. 8.
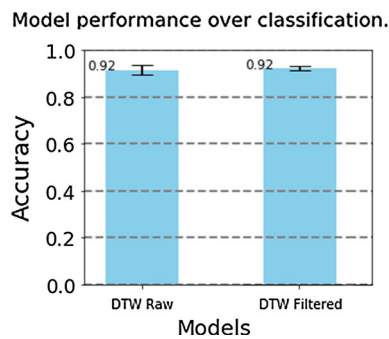
Fig. 9 The impact of raw and filtered temperature-series data on performance of k-NN-DTW model on the P_{large} dataset (3587 samples) as measured using 10-fold cross-validation

DTW performs best here with the highest average accuracy and least variance across the folds. SAX and SFA fare badly performing worse that the Euclidean distance baseline.

The impact of cleaning the signal

Accepting that DTW is the best distance measure, we move on to consider other aspects. The raw pyrometer data is quite noisy so we have passed it through a Butterworth filter for our initial analysis. It is possible that the classificaction would work equally well on the raw data so we test this hypothesis. The results are shown in Fig. 9. k-NN-DTW performs well on the raw data but given that the variance of the accuracy on the filtered signal is better we will stick with the filtered data. The filtered signals have the added advantage that they are easier to interpret visually, see Fig. 2.

Algorithm speedup

While we have good classification accuracy at 92% using k-NN-DTW, the k-NN retrieval is slow – the cross-validation tests shown in Fig. 9 take about 40 minutes to run. These results are based on 10-fold cross validation on the P_{large} dataset (3587 samples). So we test the performance of the speedup strategies described in "k-NN speedup" section. Speedup is achieved either by accepting approximate k-NN retrieval or by editing down the training dataset. The approximate k-NN retrieval is achieved by using Annoy to retrieve a large candidate set of 400 and then selecting the k neighbours from that set using DTW. By contrast, the case editing strategies (CNN-DTW and CRR-DTW) reduce processing time by removing 'redundant' samples from the training data.

The results are shown in Fig. 10. The most promising strategy proves to be Annoy-DTW. Accuracy is within 1% of full DTW but processing time is reduced to 18%. Of the instance selection techniques, CRR is the most successful but the impact of accuracy is greater than with Annoy-DTW and the speedup is not as significant.
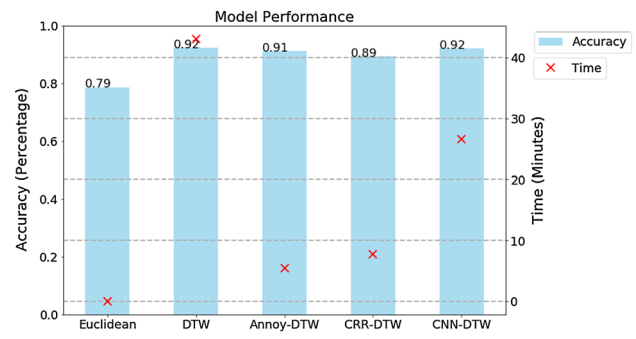

Fig. 10 The impact of speedup strategies as measured by hold-out testing on the P_{large} dataset
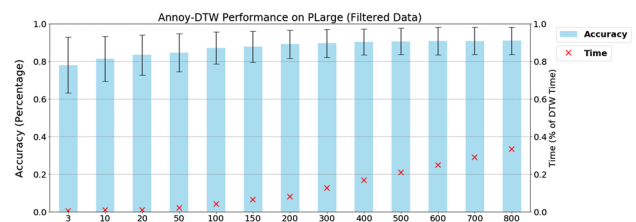

Fig. 11 Evaluation results (both accuracy and time) of Annoy-DTW model over different Euclidean neighbour sizes over P_{large} group of blocks. The time shown is the percentage of the total time of k-DTW model over the dataset

Clearly, the larger the candidate set returned by the Annoy stage the closer the Annoy DTW performance will be to exact k-NN-DTW. The downside is that the retrieval time will be impacted. This is shown in Fig. 11. We get dramatic speedup for candidate sets of size 20 or 30 but the accuracy is poor. The sweet spot seems to be with candidate sets of size 300 or 400. There is almost no loss of accuracy with retrieval times reduced below 20%.

The impact of pore size

In the final stage in our evaluation we look at the impact of pore size. The second dataset covers voids of width 0.1 mm and 0.05 mm. This is a smaller dataset with 249 pore samples and 632 non-pore. Given the smaller size of the feature it might be expected that the accuracy might not be as good as with the larger pores. However the accuracy of the Annoy DTW method is 94% measured using 10-fold cross-validation. We can conclude that the method will deal with voids of size 0.05 mm without loss of accuracy.

Conclusions

The objective of this research is to asses the use of ML for in-situ monitoring of 3D printing using PBF. The signal being monitored is the temperature of the melt pool and the monitoring is done using a k-NN classifier on the time-series data.

The core hypothesis is that anomalies such as pores can be detected by identifying characteristic signatures in melt pool time-series data. The main findings are as follows:

- These ML methods are effective for detecting these type of pores. Classification accuracies of 92% to 94% are achievable.
- The temperature data coming from the pyrometers is quite noisy so we assessed the impact of cleaning the data using a Butterworth filter. Cleaning the data in this way has little impact on classification accuracy.
- Of the three metrics evaluated DTW seems to be the most effective for detecting anomalies of this type.
- Given the computational complexity of k-NN-DTW and the need to perform classification in real-time, we have evaluated a number of speedup strategies. We find that a two-stage strategy that uses ANN to retrieve a candidate set and then exact NN search on this set is effective.

The next stage in this research is to evaluate these methods on data from a wider variety of AM builds and from anomalies arising from perturbations in the production process rather that designed artefacts.

References

Aghabozorgi, S., Shirkhorshidi, A. S., & Wah, T. Y. (2015). Time-series clustering-a decade review. *Information Systems*, *53*, 16–38.

Aminzadeh, M., & Kurfess, T. R. (2019). Online quality inspection using Bayesian classification in powder-bed additive manufacturing from high-resolution visual camera images. *Journal of Intelligent Manufacturing*, *30*(6), 2505–2523.

Aumüller, M. , Bernhardsson, E., & Faithfull, A. (2017). Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In *International conference on similarity search and applications*, (pp. 34–49). Springer

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606–660.

Bernhardsson, E. (2016). Annoy. https://github.com/spotify/annoy

Bhavar, V., Kattire, P., Patil, V., Khot, S., Gujar, K., & Singh, R. (2017). A review on powder bed fusion technology of metal additive manufacturing. In *Additive manufacturing handbook*, (pp. 251–253). CRC Press

Bose, S., Ke, D., Sahasrabudhe, H., & Bandyopadhyay, A. (2018). Additive manufacturing of biomaterials. *Progress in Materials Science*, *93*, 45–111.

Cunningham, P., & Delany, S. J. (2020). k-nearest neighbour classifiers—2nd edition. arXiv:2004.04523

Demir, A. G., Monguzzi, L., & Previtali, B. (2017). Selective laser melting of pure Zn with high density for biodegradable implant manufacturing. *Additive Manufacturing*, *15*, 20–28.

Fitzsimons, L., McNamara, G., Obeidi, M., & Brabazon, D. (2019). The circular economy: Additive manufacturing and impacts for materials processing. In *Encyclopedia of renewable and sustainable materials*, (pp. 81–92). Elsevier

Gibson, I., Rosen, D. W., Stucker, B., et al. (2014). *Additive manufacturing technologies* (Vol. 17). Berlin: Springer.

Grasso, M., Demir, A. G., Previtali, B., & Colosimo, B. M. (2018). In situ monitoring of selective laser melting of zinc powder via infrared imaging of the process plume. *Robotics and Computer-Integrated Manufacturing*, *49*, 229–239.

Grasso, M., & Colosimo, B. M. (2017). Process defects and in situ monitoring methods in metal powder bed fusion: A review. *Measurement Science and Technology*, *28*(4), 044005.

Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with gpus. arXiv:1702.08734

Keogh, E., & Kasetty, S. (2002). On the need for time series data mining benchmarks. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining-KDD 02*

Kwon, O., Kim, H. G., Ham, M. J., Kim, W., Kim, G.-H., Cho, J.-H., et al. (2020). A deep neural network for classification of melt-pool images in metal additive manufacturing. *Journal of Intelligent Manufacturing*, *31*(2), 375–386.

Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery-DMKD 03*

Mahato, V, Johnston, W., & Cunningham, P. (2019). Scoring performance on the y-balance test. In *international conference on case-based reasoning*, (pp. 281–296). Springer

Mahato, V, Obeidi, M. A., Brabazon, D., & Cunningham, P. (2020). An evaluation of classification methods for 3d printing time-series data. In *21st, IFAC World congress*

Mani, M., Lane, B. M., Donmez, M. A., Feng, S. C., & Moylan, S. P. (2017). A review on measurement science needs for real-time control of additive manufacturing metal powder bed fusion processes. *International Journal of Production Research*, *55*(5), 1400–1418.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *26*(1), 43–49.

Schäfer, P., & Högqvist, M. (2012). SFA: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International conference on extending database technology*, (pp. 516–527). ACM

Wu, F.-X., Zhang, W.-J., & Kusalik, A. J. (2005). Dynamic model-based clustering for time-course gene expression data. *Journal of Bioinformatics and Computational Biology*, *3*(04), 821–836.

Yujian, L., & Bo, L. (2007). A normalized Levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1091–1095.