ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

Quantitative and Real-Time Control of 3D Printing Material Flow Through Deep Learning

*Douglas A. J. Brion and Sebastian W. Pattinson**

3D printing could revolutionize manufacturing through local and on-demand production while enabling uniquely complex and custom products. However, 3D printing's propensity for production errors prevents autonomous operation and the quality assurance necessary to realize this vision. Human operators cannot continuously monitor or correct errors in real time, while automated approaches predominantly only detect errors. New methodologies correct parameters either offline or with slow response times and poor prediction granularity, limiting their utility. A commonly available 3D printing process metadata is harnessed, alongside the video of the printing process, to build a unique image dataset. Regression models are trained to precisely predict how printing material flow should be altered to correct errors and this should be used to build a fast control loop capable of 3D printing parameter discovery and few-shot correction. Demonstrations show that the system can learn optimal parameters for unseen complex materials, and achieve rapid error correction on new parts. Similar metadata exists in many manufacturing processes and this approach could enable the adoption of fast data-driven control systems more widely in manufacturing.

1. Introduction

Material extrusion is the most widespread 3D printing or additive manufacturing technology due to its low cost, ease of use, minimal post-processing, and compatibility with a broad palette of functional materials.[1–3] Presently, the technology is promising across a range of fields including medical devices,[4] soft robotics,[5] and building construction.[6] However, extrusion 3D printing is vulnerable to a plethora of errors that limit the adoption of these 3D printed products. These errors often arise due to the open loop nature of the manufacturing process[7–9] as currently process monitoring, parameter selection, and the application of corrections are entirely manual procedures provided by expert human operators. These are primarily determined through visual inspection and learned experience. Thus, there is considerable scope for the deployment of deep learning and computer vision techniques to automate and improve these processes. Furthermore, due to the slow manufacturing speed, it is beneficial to identify and correct defects at an early stage to avoid wasted material, energy, and time. This makes automated monitoring ideally suited. Recently, reinforcement learning techniques have been applied to control in numerous applications[10] such as robotics,[11] the electrical grid,[12] and even nuclear fusion.[13] Additionally, supervised methods have been used effectively in many fields such as autonomous vehicles with end-to-end networks enabling impressive real-time control.[14,15] These supervised approaches often utilize the latest deep convolutional neural networks and vision transformers[16–18] to determine appropriate actions such as a steering angle from input images. Limited work exists on applying such controllers to manufacturing, especially through the utilization of readily available metadata as continuous image labels. Furthermore, many existing control approaches apply large models which require considerable data and compute resources and are thus challenging to scale.

Indirect methods have been developed to detect errors during the additive manufacture of parts by monitoring acoustic emissions and printer vibrations[19,20] as well as motor current.[21,22] However, these methods often require accurate physics models in addition to expensive equipment, but are not sufficiently rich to identify or correct diverse errors. Correct extrusion creates highly repeatable and uniform patterns, and thus with vision a broad range of defects can be determined. There have been several attempts at using vision-based approaches to detect errors during the 3D printing of parts. Traditional image processing and computer vision techniques have been applied with a single camera to detect large errors such as layer shifts and poor infill.[23–26] These methods though often struggle in detecting finer scale error modes and image features. Other methods use multiple cameras to better handle occlusions and to enable 3D reconstructions of parts via techniques such as structured light scanning.[27–30] These approaches can detect a wider range of errors in greater detail. However, they are often monetarily and computationally expensive, create extra work for users with calibration steps, are sensitive to lighting conditions and part surface properties, and can be limited by scanner resolution.

D. A. J. Brion, S. W. Pattinson
Department of Engineering
University of Cambridge
Trumpington Street, Cambridge CB2 1PZ, UK
E-mail: swp29@cam.ac.uk

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202200153.

DOI: 10.1002/aisy.202200153

While the automated detection of errors is useful, expert operators are still required to make necessary online and offline parameter corrections. Recently, deep learning vision-based approaches have been used to autonomously apply corrections offline after printing for a range of errors.[31–33] These methods are very useful for errors that build over time as the material cools, such as cracking and warping, and nonrecoverable failures, for example, poor bridging. Although a significant step forward, these approaches still fail to catch many internal error modalities thanks to externally mounted cameras and result in wasted prints as corrections are applied after the fact.

Errors can be reduced further and corrected online during printing by combining vision with traditional real-time feedback loop strategies.[33–35] Object detection networks[36] have been used along with image processing techniques to localize specific errors and estimate their severity to facilitate the appropriate response.[33] However, these methods were focused on slow-growth errors and thus do not present true real-time control. Additionally, manual labeling of the training dataset was required. Real-time control of flow rate has been explored using classification methods such as k-NN[34] and ResNet models.[35] These approaches, although a large step in the right direction, currently yield slow response times. This is primarily due to the classification-based approach used in these works, where the suboptimal flow rate is either categorized as under- or over-extrusion. Corrections are thus applied with single-scale fixed increments or estimations derived from classifications, and thus large errors take numerous updates to fix. Additionally, there is a compromise between steady-state error and correction speed in choosing the appropriate step size. The slow response time is further compounded by slow sampling frequencies of 1[34] and 3.33 Hz,[35] lengthy prediction filtering strategies, and G-code execution delays. In reality, flow rate is a continuous variable and thus for real-world deployment, systems capable of estimating the precise level of flow rate are required for faster correction and improved performance.

In this work, commonly available 3D printing process metadata from firmware was harnessed, alongside a real-time video of printing, to build a new dataset of over 250 000 automatically labeled images. With this dataset, vision regression models were trained to precisely predict material flow as a continuous variable, in turn, enabling true proportional correction. We coupled this monitoring of real-time video with a new feedback loop capable of 3D printing parameter discovery and few-shot correction. This combined control system ran at sampling rates of nearly 15 Hz, over 4 times faster than the previous state of the art. Finally, the response times were further reduced through toolpath splitting and optimized prediction filtering. Experiments showed that the system can learn the optimal flow rate for unseen complex materials and achieve rapid few-shot error correction on new parts. Our method improves on prior work in 3D printing error detection, correction, and flow rate control with the application of models capable of continuous prediction for a single parameter compared to the previous discrete prediction approaches. This is achieved using a lightweight model and efficient sampling to enable significantly faster feedback speed. The data required can be rapidly collected and the models quickly trained, with the whole process from no data to deployed control system taking only 8 h total. Similar metadata exists in many manufacturing processes and this easy to deploy approach could enable the adoption of fast data-driven control systems more widely in manufacturing.

2. Results

2.1. Autolabeled Dataset Generation

A unique data acquisition system was developed to capture high-quality images of the material deposition process in extrusion AM and to label each image with metadata, specifically the current material flow rate. An endoscope camera was attached to a low-cost 3D printer to capture images of the extrusion process during printing (see **Figure 1**A). An overview of the steps in this data acquisition system is shown in Figure 1B. This pipeline enabled the creation of an entirely new autolabeled dataset of 253 405 images in less than 5 h. Sample images from this dataset alongside their labels are shown in Figure 1C. Uniquely, the approach is scalable to any number of printers, in future enabling the creation of large datasets from fleets of machines.

A range of labeled flow rate levels were required to enable the precise real-time prediction of flow rate for future unseen prints. At run time, the printer operates off relative flow rate levels

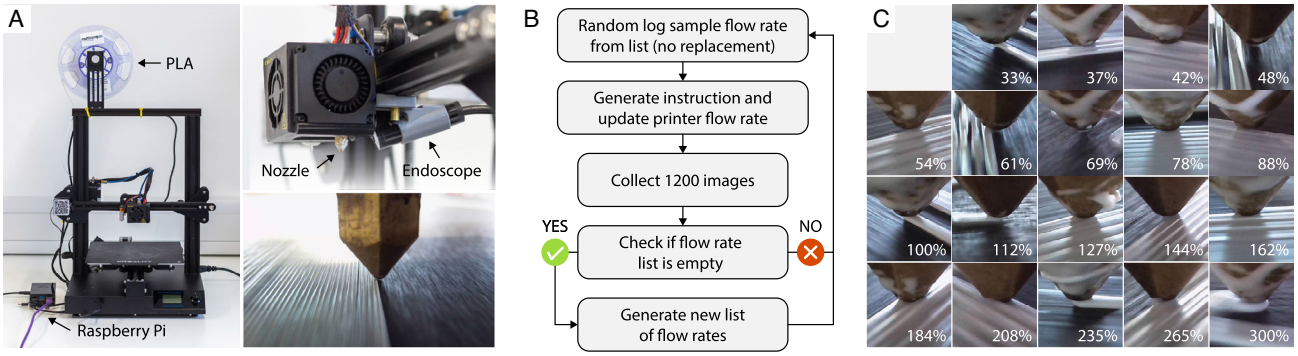

Figure 1. Data collection process for a range of flow rates across single-layer geometries. A) Creality CR-20 Pro used for data collection equipped with Raspberry Pi Model 4 B+ and endoscope camera recording video at 1080 pixel resolution. Endoscope was mounted to printer using an in-house printed clip on mount. B) Outline of data collection pipeline. 19 levels of flow rate are used in log space, mapping to a range of $\frac{1}{3}$ to 3 times optimal flow. C) Image of each flow rate level randomly sampled from the test set.

(in integer percentages), which act as multipliers for the input absolute flow rate amount specified. Upper and lower relative flow rate limits were set to three times (300%) and one-third (33%) of optimal extrusion. The natural log space of this range was sampled at regular intervals to ensure even coverage and a symmetric dataset with an equal number of over- and under-extrusion levels. In total 19 flow rate levels were used. The resolution and spacing of these levels were determined qualitatively through experimentation by printing samples at a range of flow rates centered around 100% to determine the smallest relative change with a noticeable impact on part quality. For under-extrusion, this was when gaps first appeared between the extruded paths and for over-extrusion when the extruded paths overlapped to increase surface roughness, with the optimal range resulting in completely uniform extrusion over the build surface.

In total 24 prints were completed. These prints all consisted of single-layer circular geometries with a layer height of 0.2 mm and diameter of 150 mm. All parts were printed with a hot end temperature of 205 °C, bed temperature of 60 °C, lateral speed of 45 mm s^{-1}, cooling fan enabled, and 3 external perimeters. Due to the constant speed during printing, the flow rate for all sliced parts was 3.60 mm^3 s^{-1} when using a relative flow of 100%. As such with the selected bounds, the range of flow covered was from 1.20 to 10.8 mm^3 s^{-1}. Twelve different infill types were used (2 samples printed of each). Six of the samples used 100% linear infill at 30° increments from 0° to 150°. Two used concentric infill at 25% and 100% densities. The remaining 4 prints used cross, grid, gyroid, and triangle infill patterns all at 25% density. This range of infill types was chosen to allow the vision models to generalize across different geometries.

For each print, a flow rate value was randomly sampled without replacement from a uniform distribution of 19 possible values in log space. This value was then converted back to relative percentages and sent to the printer to update the flow rate. Upon execution, 1200 images were collected and labeled for that level. Subsequently, another flow rate was sampled without replacement and 1200 images were again captured until no levels remained. A new complete set of the 19 flow rates was then generated to start the process again. With this method, a total of 253 405 labeled images were collected. In total, this only took 5 h of printing time on a single machine. With this system, the mean sampling rate across the dataset was 14.37 Hz. Overheads involved in retrieving information from the printer's firmware, capturing the snapshot, sending it over the network, and labeling the image reduced average the sampling rate from the endoscope's 30 Hz; however, this rate was still occasionally reached. Thus, the developed controller had to be capable of running faster than this maximum sampling frequency.

It was important to consider the response time to parameter changes when labeling the captured images during printing. When the relative flow rate was updated on the printer there was both a software execution delay and mechanical response time delay before the change could be visually seen. The majority of this delay comes from the mechanical response of the system as pressure needs to be increased or decreased rapidly in the hotend. To determine the maximum time taken for this delay, a series of experiments were run to go from the minimum relative flow rate of 33% to the maximum of 300% and vice versa. With the print settings used during data collection, it was determined that on average after approximately 10 s the new desired level had been reached. As such 150 images were removed after each parameter change to ensure the labeled data later used for training models was correct and did not contain any images from the transition region.

From the complete dataset, an equal number of the 19 levels of flow rate were then sampled - this number being the total samples for flow rate level with the fewest samples. This resulted in a final dataset of 125 077 labeled images which was then split through random sampling into train, validation, and test sets with a ratio of 80:10:10. Finally all images were pre-cropped to a 256 × 256 pixel region centered around the nozzle tip using the coordinates stored during collection to significantly speed up training. See Figure 1C for some examples.

2.2. Fast Flow Rate Prediction Model

2.2.1. Model Training and Performance

For the precise prediction of flow rate various sized models with a RegNet backbone[37] and single output fully connected linear layer were trained. The initial weights of these network backbones were pre-trained on the ImageNet1k dataset.[38] The models used the log space flow rates as targets - using this normalized, linear, and evenly spaced data was vital for achieving good performance.

During training, the pixel values across each red, green, and blue (RGB) channel were normalized. Additionally, several common data augmentation techniques were used as popularised on standard datasets such as ImageNet[38,39] (see **Figure 2**A). The pre-cropped and normalized 256 × 256 image centered on the nozzle tip was randomly cropped to a 224 × 224 square. The resultant cropped image was then horizontally flipped with a probability of 0.5. After these geometric augmentations, principal component analysis (PCA) color augmentation was applied to each image as popularised by AlexNet.[39] These augmentations were only applied during the training pass, and for validation passes the input 256 × 256 pixel images were center cropped to the required 224 × 224 input shape.

Five model sizes were trained for 25 epochs across 2 graphics processing units (GPUs) in parallel. The performance of these models can be seen in **Table 1**. Increasing the model size reduced the in-distribution test set loss at the cost of computation time and resource needs. Out-of-distribution (OOD) print performance appeared to be less coupled to model size, suggesting that the very large models overfitted the training data and thus they did not generalize as well. The OOD print used can be seen in **Figure 3**A and was later used for tuning averaging and filtering parameters as described in further detail in Section 2.2.2. The use of a small and fast model was just as important as the accuracy for this real-time control application, where memory footprint and the number of iterations achievable per second were important requirements. Therefore, for the remainder of the work the smallest model (RegNetY 400MF backbone) was used. Five of these models were trained using different random seeds—the mean training and validation loss along with 95% confidence intervals for these seeds can be seen in Figure 2B. These random seeds primarily affected the order in which the model saw the data and the augmentations applied, as all

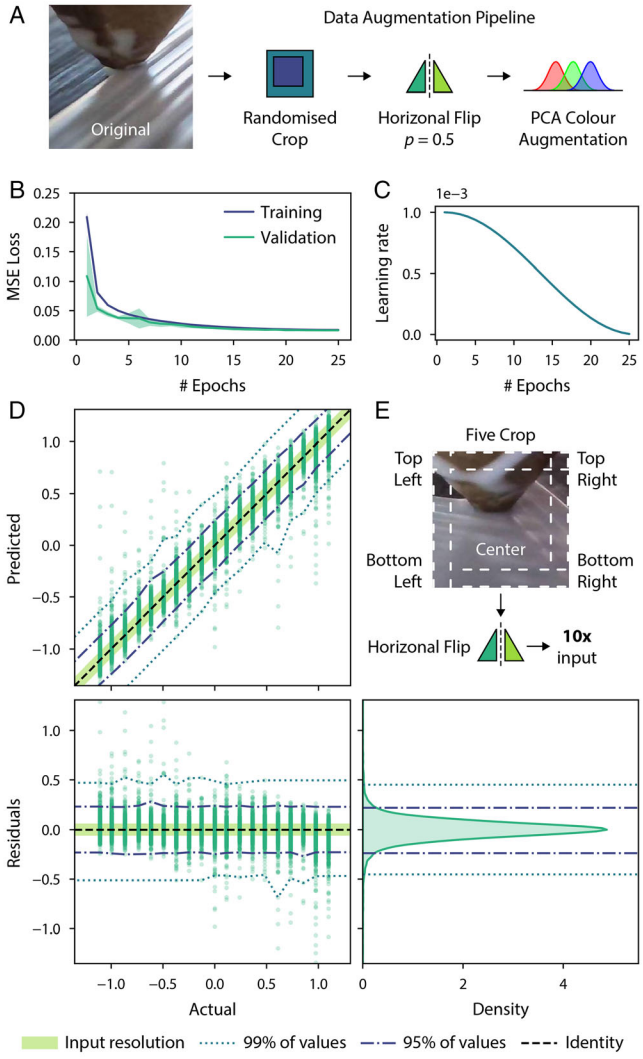

Figure 2. Training of RegNetY model and results on test set. A) Data augmentation used during the training process. Input 256 × 256 images are randomly cropped to 224 × 224 and a horizontal flip is applied with a probability of 0.5. This effectively increases our number of training samples by 2^{11}. Additionally, PCA-based RGB noise was added to the images with a standard deviation of 0.1. B) Training and validation loss for 5 random seeds of a pre-trained RegNetY-400mf model, mean is shown along with 95% confidence interval. C) Cosine Annealing Scheduler decaying the learning rate value during the training process. D) Predicted versus Actual plot on our unseen test set showing boundaries for 95% and 99% of predictions. Shaded green region is the spacing between flow rate values in our training data so represents. Residuals of the predictions and plot showing the density. E) During test time each input image is cropped in the five locations specified and mirrored creating 10 images from a single input. These are all fed through the trained network and their predictions are average using the mean.

network backbones were initialized with the pre-trained weights, with only the final output linear layer initialized differently given the seed. The mean squared error (MSE) was minimized using stochastic gradient descent and the learning rate was updated during training for all models using a cosine annealing learning rate scheduler (see Figure 2C).

Table 1. Performance of trained models with different RegNetY backbone sizes.

Model backbone	# Parameters	Iterations [s]	Test set MSE	OOD print MSE
RegNetY 400MF	3.9 M	95.8	0.0159	0.0708
RegNetY 800MF	5.6 M	97.3	0.0148	0.1003
RegNetY 1.6GF	10.3 M	56.8	0.0115	0.1262
RegNetY 3.2GF	17.9 M	44.4	0.0105	0.0878
RegNetY 8GF	37.4 M	13.8	0.0092	0.0752

At test time, to improve the predictive performance, each 256 × 256 input image was cropped into five 224 × 224 windows for the top left, top right, bottom left, bottom right, and center. Then, each of these five images was horizontally flipped resulting in 10 images produced from the single input (see Figure 2E). These 10 images were stacked together and passed through the network in a single forward pass. The 10 output predictions were subsequently averaged using the mean. Without this augmentation, when using a single center cropped image, an MSE of 0.0187 was achieved for the RegNetY 400MF backbone compared to the 0.0159 MSE with augmentation. Thus, the multi-crop approach led to a boost in performance of 14.6% at test time. We also tested each model on an out-of-distribution print and noticed a drop in performance; however, the predictions were sufficiently accurate to still enable real-time control.

The results of the small trained model on the held-back test set can be seen in Figure 2D. The trained network accurately predicted flow rate, with the light green shaded region showing the input resolution used in training for the 19 levels of flow rate. In addition, 95% and 99% intervals are shown with the network producing relatively few outliers—these can be easily removed with suitably chosen filtering procedures. The residuals of this test data are then plotted along with a fitted Gaussian distribution illustrating that the predictions are well centered. The smallest model with test augmentation achieved an inference rate of ≈100 Hz on a single GPU—significantly higher than the endoscope's 30 Hz sampling rate or the 14.37 Hz seen during data collection. As such, there is considerable scope for running multiple printers in parallel or an ensemble of models to improve predictive performance and enable uncertainty estimation.[40]

2.2.2. Prediction Filtering and Smoothing

The trained model was capable of fast and precise flow rate level predictions from a single input image on the training, validation, and unseen test sets. However, an increased robustness to noise and inaccurate predictions was required for handling OOD samples and to smooth predictions given the significant number made during a single print. To tackle this issue, an unseen OOD print of a single layer geometry was printed with 100% linear infill at an unseen angle of 15°. This print is shown in Figure 3A and the results in Table 1. Due to the temporal nature of the printing process, more information can be gained by looking at a series of images rather than single images in isolation. Therefore, a first-in-first-out (FIFO) buffer was used to determine a rolling mean prediction over a set of images. The length of this

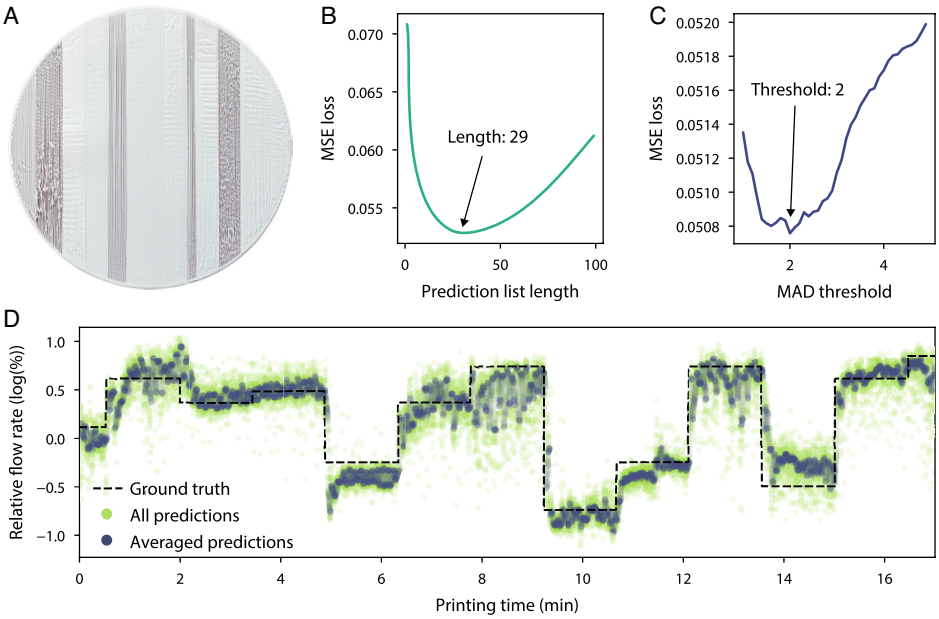

Figure 3. Using unseen print to determine optimal parameters for averaging and filtering. A) Circular test sample printed in white PLA with randomly sampled flow rate levels. B) Plot showing the effect of averaging previous predictions on performance. A rolling mean is computed using a list length of 29 predictions corresponding approximately to the past 2 s of printing. The correct list length balances the trade-off between reducing noise and still having fast response times. C) Median absolute deviation (MAD) is used to remove outliers from the prediction list. A threshold value of 2.0 provided the best results. D) Network predictions for the unseen sample shown in subfigure A. All predictions for the network are shown along with the rolling averages with list length 29, and MAD threshold of 2. Due to the rolling average, this still results in predictions at nearly 15 Hz.

FIFO had a significant impact on reducing the value of the MSE. Buffer lengths ranging from 1 to 100 predictions were tested and the MSE loss was computed (see Figure 3B). To calculate this loss, the target flow rate mean was computed and compared to the mean of the predictions in the buffer. A short FIFO was more susceptible to noise in the predictions of the network, and increasing the length of the buffer to cover just the previous second of printing resulted in an appreciable drop in error. A FIFO buffer with a long length also harmed performance as the predictions used were too far away both spatially and temporally from the current point of printing and thus the current flow rate level. The optimal length was found to be at 29 predictions, and therefore the last 29 input images. This approximately corresponded to the previous 2 s of printing.

To further improve the performance of the system, outlier predictions were removed from the buffer before the mean was computed. For this, the median absolute deviation (MAD) statistical measure was used. For the set of 29 predictions, the MAD is the absolute deviation from the median of the predictions. This can also be interpreted as the median of the absolute values of the prediction set's residuals from the median of the set. Similarly to optimizing the FIFO length, a range of MAD threshold values was swept from 1 to 5 at 0.1 intervals (see Figure 3C). Predictions with MAD values greater than this threshold were removed. A low MAD threshold, in this case, removed too many predictions and thus was a poor choice—for sparse distributions the majority of values were removed. A high threshold did not successfully remove outlier predictions, skewing the mean. A threshold value of 2 provided the best results and was selected with the FIFO length of 29. In Figure 3D the effect of these established hyperparameters for prediction buffer length and the MAD threshold is visualized. The ground truth prediction of relative flow rate is shown in log space along with all the predictions from the network. Overlayed is a tighter distribution of predictions, filtered and smoothed with the method described.

2.3. Real-World Demonstrations

2.3.1. Learning Flow Rates for Novel Materials

New materials with unknown printing parameters are continually being developed. These include foaming materials that expand upon extrusion and can be used, for example, to reduce weight or for insulation. The level of foaming achieved by these materials is directly coupled to the temperature during material deposition, with higher temperatures, in general, leading to increased foaming. The more foaming the greater the volume; thus, to achieve a uniform and dimensionally accurate prints with good surface finish the amount of material extruded must be proportionally updated to account for the expansion at different temperatures. The relationship between temperature, foaming, volume expansion, and therefore flow rate, is unknown to a new user and currently, these relationships must be found through intensive manual experimentation.

The ability to discover the correct flow rates at given temperatures for these new materials demonstrates the effectiveness of the trained model. **Figure 4**A shows a test part printed from foaming PLA with 100% relative flow ($3.60\,mm^3\,s^{-1}$) at $45\,mm\,s^{-1}$ lateral speed, with 100% linear infill, 3 outer

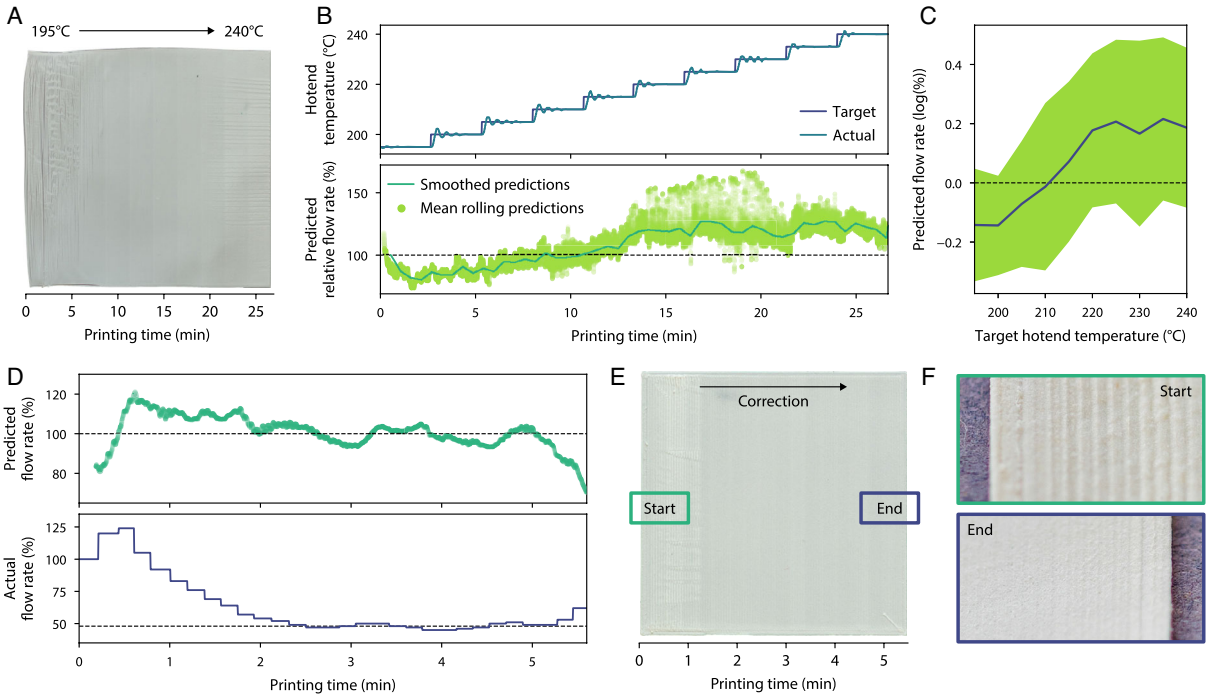

Figure 4. Self-learning the optimal flow rate for unseen foaming PLA. A) Printed sample of foaming PLA with the constant flow rate set to 100% with increasing temperature in 5 °C increments from 195 °C to 240 °C. B) Plot showing the target and actual temperature during the print alongside rolling mean relative flow rate predictions. C) Relationship of hotend temperature to predicted flow rate. The model correctly predicts that higher temperatures cause greater foaming. D) Self-determining the optimal flow rate for foaming PLA at a temperature of 235 °C. Model stabilizes at 48% relative flow (1.728 $mm^3\ s^{-1}$) providing good and consistent extrusion. E) Image of the flow rate corrected part. F) Macro shots of the start of the print at 100% flow and after reaching correct level automatically.

perimeters, 0.2 mm layer height, and 0.4 mm line width. Throughout the course of this part, the temperature was increased from 195 °C to 240 °C at evenly spaced intervals 20 mm apart (each taking approximately 2 min 45 s). There were noticeable changes in extrusion with vertical bands appearing in the print. At lower temperatures the part suffered from severe under extrusion; however, at higher temperatures, there was significant over extrusion. Plots showing the increase in temperature can be seen in Figure 4B along with the model's mean rolling flow rate predictions. This plot highlights that the model generalized to an unseen material and successfully predicted under extrusion at low temperatures, good extrusion at 215 °C, and over extrusion at higher temperatures. The predictions for each target temperature were then averaged and plotted with their standard deviations in Figure 4C. The use of such an automated learning system to determine the couplings and relationships between parameters is clear. Users could apply such systems to autonomously predict optimal parameter levels for new unseen materials in a single sample print, and subsequently, use these levels on future parts.

We go further and show that human operators are not required to manually correct parameters using the predictions of the machine learning model. The model enables rapid real-time closed-loop control, allowing the printer to self-correct and autonomously learn the optimal combination of parameters (see Figure 4D). A part was printed in the foaming PLA at 235 °C with the same settings as described previously (see Figure 4E). During the printing process, images were captured and sent over the network for inference with test time data augmentation. By taking the inverse of the predicted relative flow rate, updates were generated to proportionally correct towards the optimal level, and then sent to the printer over the network. To increase the likelihood that the steady-state prediction for this unseen material was correct and to reduce the chance of overshooting slower updates were made, with 150 predictions averaged before sending a correction. Prior to printing, the G-code toolpath of the object was split into subpaths with a maximum length of 1 mm to reduce the firmware response time for updates. G-code commands are executed sequentially, thus without splitting long moves result in significant correction delays. The plots in Figure 4D show the predicted flow rates and the updates made to the actual flow rate until the steady state was achieved. Figure 4E shows the part in its entirety—changes in extrusion level are visible, especially between the 1 and 2 min markers. During the main printing sequence from 2.5 to 5 min, the closed-loop system was remarkably stable and kept the flow rate within a tight range. The very beginning and very end of the predictions in Figure 4D show detection of under extrusion. During the initial lines of printing predicting, the level of extrusion is challenging due to the lack of interaction with adjacent paths. This is compounded by the current bias within the training dataset, as there are more images of the dense infill than the initial paths. Nozzle images at the end of printing appear as under extrusion, because the bed is visible between deposited

paths - the primary feature of under extrusion. As such, greater training data of these final lines is required for accurate predictions; however, this current limitation was acceptable as only the final seconds of printing were affected causing minimal impact. Figure 4F shows close-up images at the start of the print with a 100% flow rate at 235 °C and at the end with an optimal steady-state flow rate of 48%. These images show a clear improvement in print quality with minimal overshooting and oscillation.

2.3.2. Rapid One-Shot Online Correction

For the production of end-use parts, control systems must correct errors rapidly to minimize their impact on dimensional accuracy and mechanical performance. Being able to precisely predict the exact distance from optimal extrusion allows the developed system to correct errors in one or very few actions. Due to the sampling rate and split toolpath, this update can be applied rapidly after an error occurs.

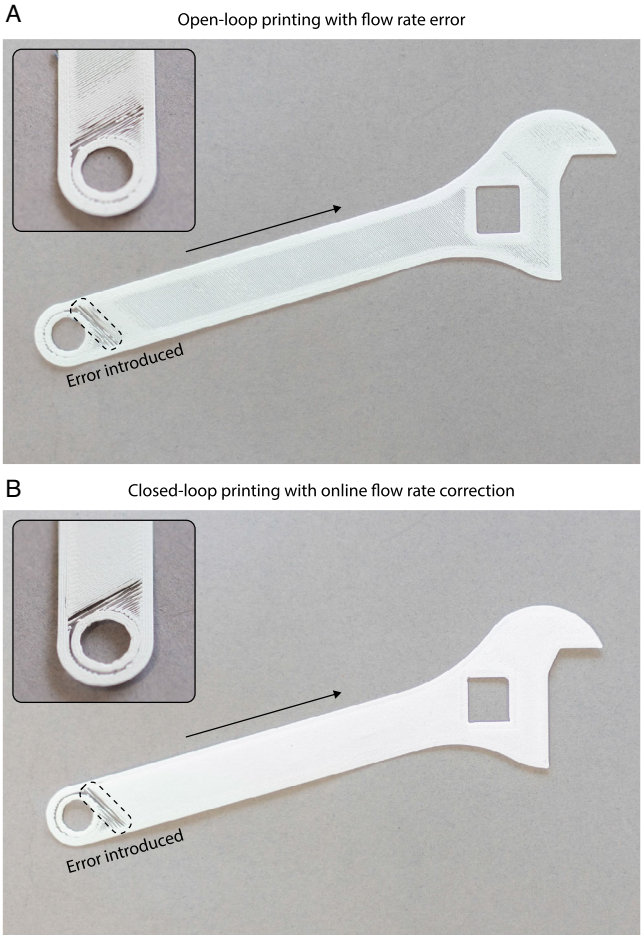

Figure 5. Demonstration of rapid online flow rate correction in the first layer of an adjustable spanner. A) Control sample with an error introduced, reducing the flow rate to 50%. After the error, the remainder of the print was severely under-extruded. B) Correction sample with an error introduced, reducing the flow rate to 50%. The few-shot control system detected the error and accurately predicted the required correction.

In **Figure 5**A, the first layer of a spanner geometry is shown, with the overall printing direction indicated by the black arrow. An error was introduced reducing the flow rate to 50% of optimal (3.60–1.80 $mm^3 s^{-1}$). No feedback or automated correction was applied, thus, significant under extrusion is visible for the remainder of the print. The print settings were the same as used for previous prints in this work with a layer height of 0.2 mm, hotend temperature of 205 °C, bed temperature of 60 °C, lateral speed of 45 $mm s^{-1}$, cooling fan enabled, and 3 external perimeters. A second print was then run with identical settings but with error correction enabled. A FIFO buffer of length 29 and a MAD threshold of 2 were used as described previously. Figure 5B shows that the system was able to accurately predict and correct the flow rate rapidly. Unlike previous work, the controller does not iteratively approach optimal flow, but jumps in one- or few-shots to the correct level. The controller showed good steady-state behavior with no large incorrect updates applied; however, minor oscillations did sometimes occur. To mitigate this, the response time was updated depending on the flow rate prediction made. For predictions near 100%, updates are less time critical and a greater level of accuracy was required. Therefore, for FIFO predictions between 89% and 113%, the average was taken across 20 buffers before an update was made. Also to stop overshooting caused by the mechanical and software response time of the system, after very large updates a delay of 150 images was used to ensure that new predictions were only made after the previous update had been realized. This delay was deemed to be acceptable due to the one-shot nature of our control algorithm meaning often a single update was all that was required.

3. Conclusion

Here we report a real-time closed-loop control system for 3D printing which is capable of predicting the precise level of material flow rate enabling parameter discovery for unseen materials and rapid few-shot correction. Commonly available 3D printing parameter information was harnessed, alongside the video of the printing process, to autonomously create a new dataset of over 250 000 labeled images. With this dataset, multiple regression models were trained to precisely predict the level of relative material flow as a continuous variable from a single input image. The temporal nature of extrusion printing was utilized by combining multiple predictions to further improve the accuracy of the system. The trained model was then used on complex materials to predict and learn the relationship between printing temperature and material flow rate in both an offline and online fashion, enabling autonomous parameter discovery for unknown materials - a previously manually intensive and tedious process. Also, the same model was used as a few-shot controller to rapidly correct flow rate and recover parts after severe errors were introduced. Due to our unique data generation and labeling procedure of combining readily available manufacturing metadata with video the system was even capable of correcting errors in a single action. Importantly, the aforementioned was achieved in a short time, ≈8 h, with data collection taking only 5 h on a single printer and the lightweight model taking 3 h to train. This speed of deployment to new settings, alongside the use of metadata that

is available for many manufacturing processes, may aid industry uptake and the potential future applications of data-driven control.

There is significant potential to build upon various aspects of this work. The current dataset is focused on the initial layer of printing and was collected using one material on one machine and had a bias toward densely filled parts. A larger and more diverse dataset is important to achieve good generalization across systems. Expanding the problem space to include more layers and infill types would increase data collection times. However, this could be addressed with parallelization, using a scalable fleet of printers to collect data simultaneously, and with intelligent sampling, to obtain the most informative data within the problem space. The algorithm and process described in this work could also generalize to a range of parameters other than flow rate (e.g., hotend temperature, Z offset, and cooling fan speed) if provided with the appropriate data. No alteration of the system would be required for parameters that present themselves in optical images. However, a wider range of parameters could be covered with the addition of further sensors such as accelerometers and infra-red cameras. Furthermore, image-based models were chosen to achieve the fast training times in this work. However, moving toward video approaches[41,42] which combine spatial and temporal information would likely be beneficial and improve model performance. The printer side of the control loop should be developed further to increase the sampling rate to enable even faster response times as this is the current bottleneck. This could be achieved by not retrieving flow rate information with every image captured, upgrading the endoscope and Raspberry Pi, and replacing OctoPrint with a less computationally intensive alternative.

Finally, this work reinforces the need for both predictive and in-situ solutions to 3D printer control and optimization. Even with future developments in corrective systems, errors will still be present in parts due to processing times and machine limitations. This could result in failed prints depending on the design requirements of the part. As such, there is scope for using monitoring systems like the one presented in this work to train predictive and preventative models capable of determining the likelihood of an error prior to printing.

4. Experimental Section

Equipment Setup: The hardware used in the setup consisted of a 3D printer (Creality CR-20 Pro) equipped with a low-cost USB endoscope (Pancellent END-AU-108A) capable of recording 5 megapixel video at a frame rate of 30 Hz. The endoscope was mounted to the printer using a custom-designed and 3D-printed mount, which required no fastenings and as such could be attached to the print head carriage with ease. The positioned endoscope focussed on the nozzle tip and thus the latest material deposition from the 0.4 mm diameter nozzle on the printer. This endoscope was connected over USB to a Raspberry Pi 4 Model B+, which in turn was connected to the 3D printer over a USB serial connection. This Pi was running a Raspbian-based distribution and an OctoPrint server with a custom data collection plugin. The in-house plugin captured 1920×1080 pixel snapshots from the endoscope at regular intervals. Each snapshot captured was then labeled with the current relative flow rate of the printer, this value was retrieved directly from the printer's firmware, in this case, Marlin version 1.1.9, using the M221 G-code command. The printer ran a configured version of Marlin with electrically erasable programmable read-only memory (EEPROM) chit-chat enabled as well as features such as thermal runaway protection to ensure safety when printing unattended. The plugin, after pairing each captured image with its respective relative flow rate, subsequently sent this information over the network to a custom server for storage. This server application saved each incoming image and created a CSV file for each print containing the relative flow rates, images, timestamps, and nozzle tip coordinates in the image.

Model Training: The models were written in PyTorch[43] and trained on two Nvidia Quadro RTX 5000 GPUs with an i9-9900 K CPU (8 cores 16 threads) and 64 GB of random-access memory (RAM). This setup was used for real-time correction but with only a single GPU. The geometric augmentation applied with cropping and horizontal flipping effectively increased the size of the dataset by a factor of 2^{11}. For the color augmentation, PCA was performed on the RGB pixel values for the cropped images in the full dataset prior to training. During training multiples of the three principle component eigenvectors found were proportionally added to each image pixel across RGB channels corresponding to each of their eigenvalues multiplied by a random variable drawn from a zero mean Gaussian distribution with a standard deviation of 0.1. For training the model, the mean squared error (MSE) loss was minimized using a stochastic gradient descent optimizer with an initial learning rate of 1×10^{-3}, momentum of 0.9, and weight decay of 5×10^{-5}. A batch size of 8 was used throughout training—small batch sizes resulted in faster convergence.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This work has been funded by the Engineering and Physical Sciences Research Council (EPSRC) Ph.D. Studentship EP/N509620/1 to Douglas Brion, Royal Society award RGS/R2/192433 to Sebastian Pattinson, Academy of Medical Sciences award SBF005/1014 to Sebastian Pattinson, Engineering and Physical Sciences Research Council award EP/V062123/1 to Sebastian Pattinson, and An Isaac Newton Trust award to Sebastian Pattinson. For the purpose of open access, the author has applied a Creative Commons Attribution (CC-BY) license to any Author Accepted Manuscript version arising.

Conflict of Interest

Douglas Brion owns a company in the area of AM error detection. Douglas Brion and Sebastian Pattinson are inventors on a patent application covering the contents of this report.

Author Contributions

D.B.: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition; S.P.: Conceptualization, Methodology, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

Data Availability Statement

The data that support the findings of this study are available in the supplementary material of this article.

Code Availability Statement

The code used to generate the results in the article will be released in a public GitHub repository on the Complex Additive Materials Group GitHub organisation page (https://github.com/cam-cambridge). Neural networks were developed with PyTorch v1.10.0 (https://github.com/pytorch/pytorch), Torchvision v0.11.1 (https://github.com/pytorch/vision), PyTorch Lightning v1.5.10 (https://github.com/PyTorchLightning/pytorch-lightning), Kornia v0.6.3 (https://github.com/kornia/kornia), Data analysis used Python v3.6.9 (https://www.python.org/), NumPy v1.19.5 (https://github.com/numpy/numpy), Pandas v1.1.5 (https://github.com/pandas-dev/pandas), SciPy v1.5.4 (https://www.scipy.org/), Seaborn v0.11.2 (https://github.com/mwaskom/seaborn), Tensorboard v2.8.0 (https://github.com/tensorflow/tensorboard), Matplotlib v3.3.4 (https://github.com/matplotlib/matplotlib), Jupyter v1.0.0 (https://jupyter.org/), JupyterLab v.2.2.9 (https://github.com/jupyterlab/jupyterlab), Pillow v8.4.0 (https://github.com/python-pillow/Pillow). Data collection and parameter correction servers were developed with Flask v1.1.1 (https://github.com/pallets/flask), OctoPrint v1.6.1 (https://octoprint.org/), Marlin 1.1.9 (https://marlinfw.org/). Slicing and print preparation used Cura v4.13.1 (https://github.com/Ultimaker/Cura).

Keywords

3D printing, additive manufacturing, closed-loop control, computer vision, error detection and correction, machine learning

Received: June 3, 2022
Revised: July 11, 2022
Published online:

[1] E. MacDonald, R. Wicker, *Science* **2016**, *353*, 2093.
[2] T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Nguyen, D. Hui, *Composites, Part B* **2018**, *143*, 172.
[3] A. Cano-Vicent, M. M. Tambuwala, S. S. Hassan, D. Barh, A. A. Aljabali, M. Birkett, A. Arjunan, Á. Serrano-Aroca, *Addit. Manuf.* **2021**, *47*, 102378.
[4] S. W. Pattinson, M. E. Huber, S. Kim, J. Lee, S. Grunsfeld, R. Roberts, G. Dreifus, C. Meier, L. Liu, N. Hogan, A. J. Hart, *Adv. Funct. Mater.* **2019**, *29*, 1901815.
[5] A. Kotikian, R. L. Truby, J. W. Boley, T. J. White, J. A. Lewis, *Adv. Mater.* **2018**, *30*, 1706164.
[6] S. C. Paul, G. P. A. G. van Zijl, M. J. Tan, I. Gibson, *Rapid Prototyp. J.* **2018**, *24*, 784.
[7] B. M. Colosimo, Q. Huang, T. Dasgupta, F. Tsung, *J. Qual. Technol.* **2018**, *50*, 233.
[8] B. Searle, D. Starkey, *J. Med. Radiat. Sci.* **2020**, *67*, 43.
[9] R. Song, C. Telenko, in *Proc. 27th Annual Int. Solid Freeform Fabrication Symp.*, University of Texas at Austin, Austin, Texas, USA **2016**, pp. 1217–1229.
[10] R. Hafner, M. Riedmiller, *Mach. Learn.* **2011**, *84*, 137.
[11] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, S. Levine, in *2019 Int. Conf. on Robotics and Automation (ICRA)*, Montreal, Canada **2019**, pp. 6023–6029, https://doi.org/10.1109/ICRA.2019.8794127.
[12] J. Wang, W. Xu, Y. Gu, W. Song, T. C. Green, *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 3271.
[13] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, et al., *Nature* **2022**, *602*, 414.
[14] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, arXiv 1604.07316, **2016**.
[15] A. Prakash, K. Chitta, A. Geiger, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, IEEE, Piscataway, NJ **2021**, pp. 7077–7087.
[16] K. He, X. Zhang, S. Ren, J. Sun, in *2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, IEEE, Piscataway, NJ December 2016, pp. 770–778, ISBN 978-1-4673-8851-1, http://ieeexplore.ieee.org/document/7780459/.
[17] M. Tan, Q. Le, in *Proc. of the 36th Int. Conf. on Machine Learning*, Vol. *97*, PMLR, Long Beach, California, USA **2019**, pp. 6105–6114, https://proceedings.mlr.press/v97/tan19a.html.
[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, in *9th International Conference on Learning Representations (ICLR) 2021*, Virtual Event, Austria, May 3–7 **2021**, https://openreview.net/forum?id=YicbFdNTTy.
[19] Y. Tlegenov, G. S. Hong, W. F. Lu, *Robot. Comput.-Integr. Manuf.* **2018**, *54*, 45.
[20] H. Wu, Z. Yu, Y. Wang, *Int. J. Adv. Manuf. Technol.* **2017**, *90*, 2027.
[21] Y. Tlegenov, W. F. Lu, G. S. Hong, *Prog. Addit. Manuf.* **2019**, *4*, 211.
[22] C. Kim, D. Espalin, A. Cuaron, M. A. Perez, E. MacDonald, R. B. Wicker, *2015 IEEE Int. Conf. Advanced Intelligent Mechatronics (AIM)*, IEEE, Piscataway, NJ August 2015, ISBN 978-1-4673-9107-8, http://ieeexplore.ieee.org/document/7222632/.
[23] J. Straub, *Machines* **2015**, *3*, 55.
[24] K. He, Q. Zhang, Y. Hong, *J. Intell. Manuf.* **2019**, *30*, 947.
[25] Y. Wang, J. Huang, Y. Wang, S. Feng, T. Peng, H. Yang, J. Zou, *IEEE/ASME Trans. Mechatron.* **2020**, *25*, 2287.
[26] T. Huang, S. Wang, S. Yang, W. Dai, *J. Intell. Manuf.* **2021**, *32*, 2181.
[27] O. Holzmond, X. Li, *Addit. Manuf.* **2017**, *17*, 135.
[28] M. Preissler, C. Zhang, M. Rosenberger, G. Notni, in *2018 Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, Piscataway, NJ **2018**, pp. 1–6, ISBN 978-1-5386-6602-9, https://ieeexplore.ieee.org/document/8615803/.
[29] J. Fastowicz, M. Grudziński, M. Tecław, K. Okarma, *Entropy* **2019**, *21*, 97.
[30] P. Charalampous, I. Kostavelis, C. Kopsacheilis, D. Tzovaras, *Int. J. Adv. Manuf. Technol.* **2021**, *115*, 3859.
[31] J. M. Gardner, K. A. Hunt, A. B. Ebel, E. S. Rose, S. C. Zylich, B. D. Jensen, K. E. Wise, E. J. Siochi, G. Sauti, *Adv. Mater. Technol.* **2019**, *4*, 1800653, see reference 2.
[32] M. V. Johnson, K. Garanger, J. O. Hardin, J. D. Berrigan, E. Feron, S. R. Kalidindi, *Addit. Manuf.* **2021**, *46*, 102191.
[33] D. A. J. Brion, M. Shen, S. W. Pattinson, *Addit. Manuf.* **2022**, *56*, 102838.
[34] C. Liu, A. C. C. Law, D. Roberson, Z. J. Kong, *J. Manuf. Syst.* **2019**, *51*, 75.
[35] Z. Jin, Z. Zhang, G. X. Gu, *Manuf. Lett.* **2019**, *22*, 11.
[36] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, in *Proc. of the IEEE Computer Society Conf. Computer Vision and Pattern Recognition 2016*, IEEE, Piscataway, NJ December 2016, p. 779.
[37] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, P. Dollar, in *2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, IEEE, Piscataway, NJ **2020**, pp. 10425–10433, ISBN 978-1-7281-7168-5, https://ieeexplore.ieee.org/document/9156494/.
[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, in *2009 IEEE Conf. Computer Vision and Pattern Recognition*, IEEE, Piscataway, NJ **2009**, pp. 248–255, ISBN 978-1-4244-3992-8, https://ieeexplore.ieee.org/document/5206848.

[39] A. Krizhevsky, I. Sutskever, G. E. Hinton, *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1106.

[40] B. Lakshminarayanan, A. Pritzel, C. Blundell, in *Proc. 31st Int. Conf. Neural Information Processing Systems, NIPS'17*, Curran Associates Inc., Red Hook, NY, USA **2017**, pp. 6405–6416, ISBN 9781510860964.

[41] C. Feichtenhofer, in *2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, IEEE, Piscataway, NJ **2020**, pp. 200–210.

[42] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic, C. Schmid, in *2021 IEEE/CVF Int. Conf. Computer Vision (ICCV)*, IEEE, Piscataway, NJ **2021**, pp. 6816–6826.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, in *NeurIPS* (Eds: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, R. Garnett), Vancouver, Canada, **2019**, pp. 8024–8035.