AI integrated 3D Printer Monitoring Solution

Jithin Rajan Varghese
WBooth School of Engineering Practice and Technology
McMaster University
Hamilton, Canada
jithinrajanvarghese@gmail.com

Xuanyu (Liam) Liu
WBooth School of Engineering Practice and Technology
McMaster University
Hamilton, Canada
liamliuxy@outlook.com

***Abstract*—This paper explores implementing AI-based machine vision on edge devices for inference to create alert mechanisms for 3D print monitoring jobs. This project's current scope is identifying defects via a Pi cam module attached to a Raspberry Pi and allowing the user to perform remote monitoring only when human intervention is required. To do this, we use an interactive bot that can be used to select features available from the Pi remotely. The system was designed with expansion in mind, adding modules and improving functionality. The current system has temperature and humidity, camera angle, real-time camera footage, and AI image monitoring supported. It also features login restrictions for authorized users and encryption of passwords and usernames with no on-premise storage of unencrypted data.**

***Index Terms*—3D Printer, Raspberry Pi, Inception V3, YOLO-v8, Telegram Bot**

I. Introduction

3D printing technology has brought much convenience to the modern-day production industry. Through the printing process, defects are commonly observed by the user. Having to monitor the 3D printing process with full-time human supervision drastically decreases the working efficiency. Therefore, a robust remote monitor solution is greatly compatible with the user's current dilemma.

This paper discusses the combination of hardware and software, primarily working with Raspberry Pi with AI monitoring tools, including Inception V3 and YOLO-v8, to perform defect prediction and message return through a telegram conversation bot. The tool aims to provide real-time monitoring services catering to personal 3D printing usage.

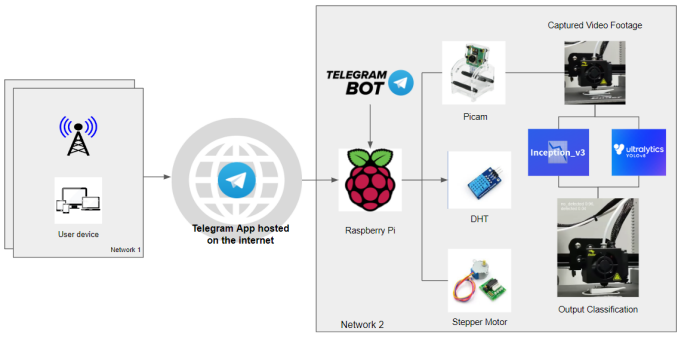

Fig. 1. Overall connection diagram.

II. Project Architecture

As shown in Fig. 1, the project's overall design is split into 2 parts: the software and hardware sides. On the hardware front, we connect Raspberry Pi with Pi Cam, DHT11 sensor, and a stepper motor. additionally, we had a ribbon cable for connecting the pins from Pi to a breadboard with a power test LED circuit to ensure the connections were set properly. On the software front, we set up the Pi with an environment management tool, "Mambaforge," and created an environment with Python 3.9 and TensorFlow 2.14. These compatible versions could run the required inference and load the dependent packages. We have a few significant parts to the code, the major ones being the telegram bot code, inference, the camera angle algorithm, and encryption that was built to ensure user privacy is met. The code on execution is meant to run for a single user but can be expanded with proper resource management methods.

III. Solution Details

A. Hardware

In this project, the hardware component is assembled primarily with the edge device Raspberry Pi; readings are extracted from multiple sensors, including DHT11 sensors and the Pi Camera. Actuators are installed below the camera to provide angular rotation to control the degree of the camera relative to the 3D printer. More details related to the hardware are mentioned below:-

1) Raspberry Pi: As shown in Fig. 3, the edge device for core computation used in this project is Raspberry Pi 4 2018 Model B, as the main logic component of the project with a SanDisk © Edge 32GB Class 10 Micro-SD serving as the storage module for the Raspberry Pi 4 Model B.

2) DHT11: The DHT11 temperature and humidity sensor shown in Fig. 3, [1] is utilized in this project to collect temperature data ranging between 0°C to 50°C and humidity between 20% to 90%. With an expected error of 2°C and 5% respectively [2] [3]. The reading from the DHT11 sensor is used to analyze the relativity between room temperature and humidity associated with the adhesion for the 3D printer build plate.

3) Stepper Motor: As shown in Fig. 4, the Pi Camera is mounted over a 28BYJ-48 Stepper Motor driven by a ULN2003AN control board. The stepper motor takes input

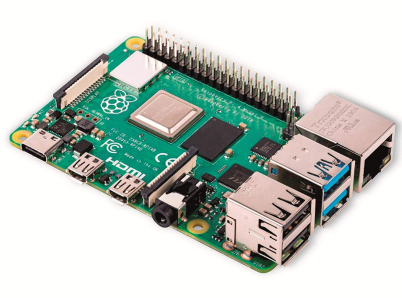

Fig. 2. Raspberry Pi 4 Model B.

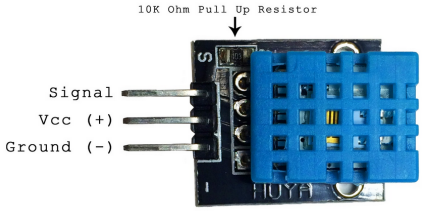

Fig. 3. DHT11 module [4].

voltage ranging between 5 to 12 volts [5]. This motor is used to control the angular movement of the camera to toggle the degree of the camera to work with the 3D printing process.

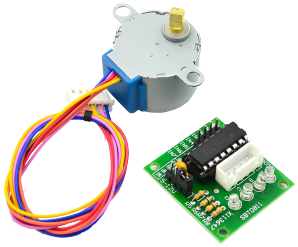

Fig. 4. Stepper motor control and board [6].

4) Pi camera: As shown in Fig. 5, the Raspberry Pi Camera Revision 1.3 module captures video/images for our project. The output from the camera is further analyzed by the AI monitoring tool to predict and classify it into defect or non defect.

5) Auxiliary components: The auxiliary components are an LED board power check and a GPIO header out board that allows connection to a breadboard for further connecting more hardware and supporting all the before-mentioned sensors except for the Pi cam which is connected directly to the Raspberry Pi.

B. Software

1) Telegram Bot:

a) Camera Angle Selector: The angle selector algorithm functions with memory of its current state. So the movement is made to turn to corresponding angles input, say 30° and the

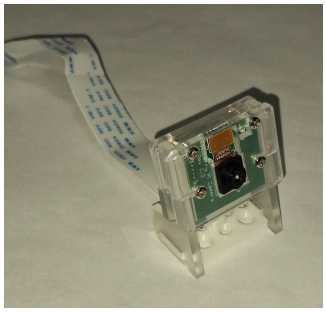

Fig. 5. Pi Camera with 3D printed mount.

Fig. 6. LED breadboard power check.

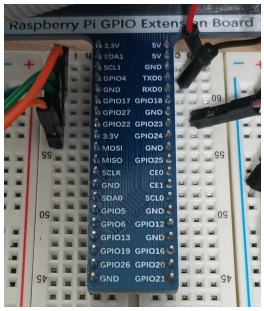

Fig. 7. GPIO extension board with cable.

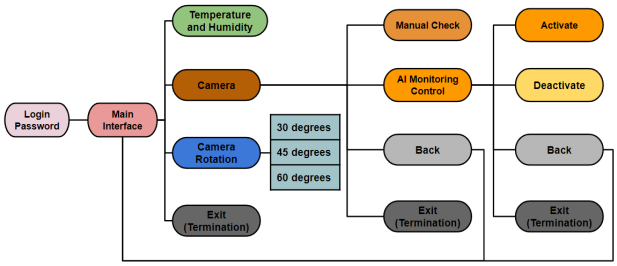

Fig. 8. Telegram algorithm flow.

following input is made for 45° The algorithm calculates the difference in angle and turns 15° more, and if the subsequent entry is 45° the algorithm will move to the initial position this, allows movement from any state to any other without moving out of the operational range of 0° to 60° with pre-selections available for 30°, 45° and 60°.

b) Encryption and environment variables: The environment variable is set up for keeping the required tokens and passwords in a secure machine-specific OS and then loaded into the program while executing. This keeps the encrypted values securely away from third-party attackers. The encryption function used is Password-Based Key Derivation Function 2 with HMAC and SHA 256 [7]. This allows for high security to be maintained with no open passwords or any individual's username being stored online.

c) Multiprocessing and Threads: The code run on the PI is split into 2 Processes: the inference code runs on the main thread, and the telegram bot is executed on a branch thread. This allows for parallel execution of code and has to share data between the 2 threads to take the full capability of the Pi. The reason why inference was run on the main thread was that it was observed the "model.predict" command from the Keras module would not operate on a branch thread. This was an interesting finding.

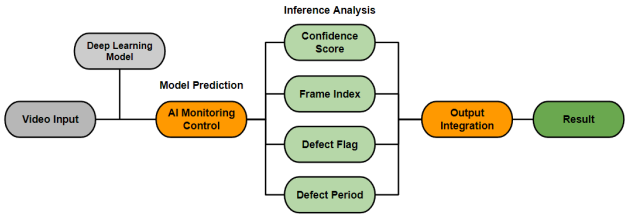

Fig. 9. AI monitoring tool control diagram.

2) AI Monitoring tool:

a) Dataset Description: The first model designed for this project utilized a labeled data-set from Kaggle [8]. This 3D printer defect classification data-set contains various images of 3D printed objects from seven different 3D printers with defects and no defects. The 3D printer defect classification data-set contains images with multiple attributes with folder-level classification to distinguish mainly between defect vs. non-defect. For the second part of this project, we considered a larger data-set [9] due to the shortcomings of the initial data-set. The newer data-set was from a Kaggle competition to determine early defect-recognition in 3D print jobs. The extensive data-set has 11.6 gigabytes of image data comprising over 100,000 images [9]. The images are annotated as defective and non-defect.

b) Data Pre-processing and Augmentation for YOLO-v8: For the initial stage of the project, the team utilizes the Roboflow open-source tool, which specializes in pre-processing and augmenting the data-set for classification assignment with the YOLO model [10] [11]. Labels are assigned in the folder structure to distinguish between classification objectives. Pre-processing solutions are introduced to decrease training time and increase performance for the entire data-set. Further augmentation is implemented to expand training data for the YOLO model to learn by generating augmented versions of each image in the training set. This includes an adjustment in shape and color and adding noisy pixels. 3,737 images are generated as an outcome of the above procedure. Cache data is introduced to the content of Google Colab through Roboflow API to work with the YOLO-v8 model. Furthermore, the team adopted the expansion competition dataset from Kaggle to generalize the data.

c) Data Pre-processing and Augmentation for InceptionV3: As part of data pre-processing for InceptionV3, we use the ImageDataGenerator function as a way to re-scale, rotate, zoom, shear, and flip the image; we also resize the image to 150*150 in RGB for use in training and validation. We also need to set up test and validation folders to be fed to the inception pre-trained model for fine-tuning. For the test part of the code, we use "cv2" to extract frames and move some of the extracted images to train and validation folders. The train and validation folder have sub-folders such as defect and non-defect with each of them. We have 64979 image files in the training set and 16323 in the validation set. In the test set, we have 174 images. The training and validation data is split in the Pareto (80:20) principle and comes from a single training ".csv" file containing image paths and defect flags. Also, we implement random sampling to split the train data into train and validation data sets. This is considered a good approach as we have quite a large number of data points.

IV. Model Development

A. YOLOv8

In the development of the 3D printing defect detection system. We selected YOLOv8 (You Only Look Once version 8) as our computer vision algorithm to detect 3D printing defects [12]. We chose the YOLO algorithm because it has photographic/video classification and can be used to classify whether there are defects during the 3D printing process.

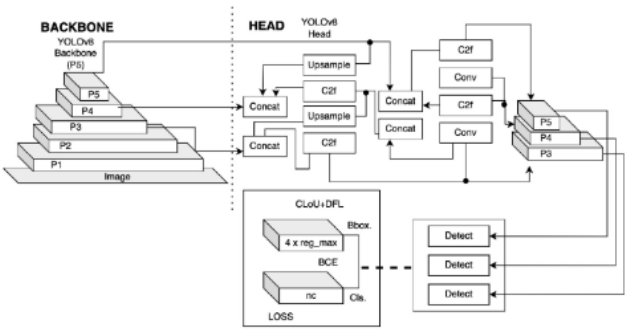

Fig. 10. YOLO-v8 Architecture.

As an iterative improvement version of the YOLO model, YOLOv8 replaced the C3 with the C2F module to restrategize

the algorithm for resolution analysis. Instead of parsing images into 2 sections with half-applied convolution, C2F extracts feature components starting from the low-resolution area and gradually moving towards the high-resolution area over images. Resulting in fewer parameters and higher processing speed over the COCO standard testing data-set. In this project, the team selected YOLO-v8n for the training.

B. Inception V3

As practically proved, large-size filters are suitable for extracting the information distributed more globally, and lower-size filters perform well when distributed locally. In the case of the identification of a defect in the 3D object, the size of the location of the information to be detected varies from image to image. So, it becomes difficult to determine the size of filters responsible for extracting the desired information from an image. To address this issue, we used the inception model that extracts feature maps by running the convolution operations using different size filters (1x1, 3x3, 5x5) along with max pooling at the same level to extract all information irrespective of its size of location.

For the application of defect identification in 3d objects, we have trained the most advanced version of the inception model that consists of core inception block at various levels but along with factorized convolution [13] [14]. Its architecture is given below.

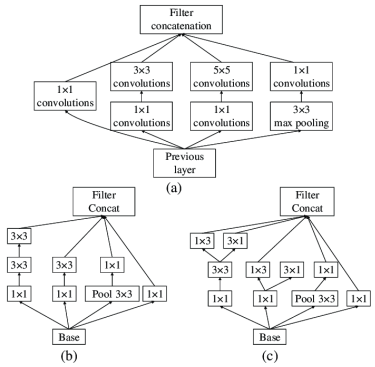

Fig. 11. Architecture of Inception V1, V2 and V3 [15].

The InceptionV3 architecture has been chosen as a computer vision algorithm for binary classification, distinguishing between defective and non-defective 3D prints. The decision to embrace Inception is because of its robust image recognition capabilities, a crucial aspect for accurately classifying defects or no defects during 3D printing. Much like the rationale for YOLOv8, Inception proves invaluable for real-time defect detection due to its efficient single-pass processing. This feature aligns seamlessly with the dynamic nature of 3D printing, allowing for swift and accurate predictions in real-time.

We trained the InceptionV3 model using a data-set from Kaggle's competition repository. It consists of 80000 images that were enough for training, validation, and testing a more generalized and accurate inception model for defect identification into 3D objects to be printed.

After preprocessing steps like re-scaling, flipping, and rotation of images using ImageDataGenerator, we trained the InceptionV3 model for 35 epochs and 140 steps per epoch.

V. Results

A. YOLOv8

Hyperparameter initialization for the YOLOv8 model assigned 10 epochs, 8 batches, and a 0.01 learning rate, with tuning improvement of the SGD optimizer to preserve the learning rate with an additional update for 0.9 in momentum. The regularization introduced is 0.3 dropout to counter the over-fitting issue. Functional parameters include categorical cross-entropy loss and sigmoid, ReLU as activation function.

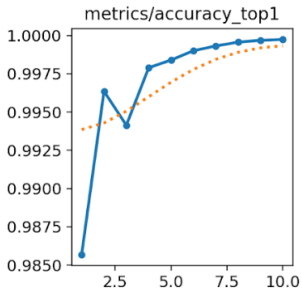

Fig. 12. YOLO training accuracy curve.

1) Accuracy and Confusion Matrix: As shown in Fig. 2, the YOLOv8 model achieves high accuracy in classifying the primary class correctly after an initial stabilization period. In addition, as shown in Fig. 3, the training loss for YOLOv8 slowly converges based on the conservative update from the SGD optimizer, suggesting that the model has nearly reached full convergence on the training data. Slight fluctuation is observed at the early stage. Overall, the change in loss is still in the expected value.

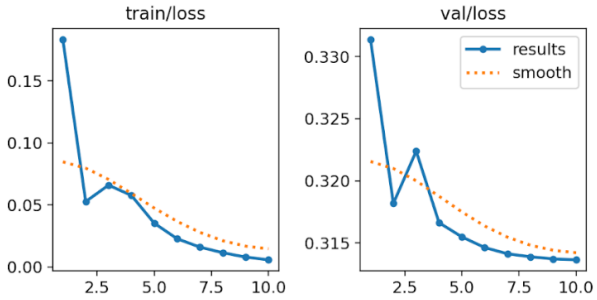

Fig. 13. Train and test loss comparison.

As shown in Fig. 4, the confusion matrix indicates a high true positive and negative. It usually indicates that the false negative rate is very low, meaning the model is highly reliable for this particular task over the test set. Nevertheless, the application is aimed at real-life printing scenarios. The lack of generalization and over-fitting issues could still result in inaccurate results.

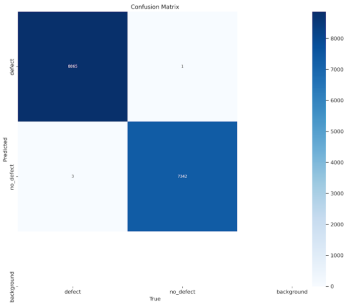

Fig. 14. Normalized confusion matrix for YOLOv8 model.

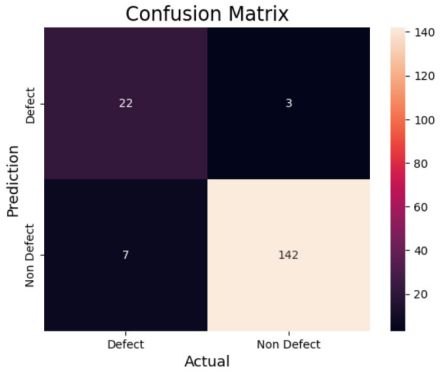

Fig. 16. Confusion matrix of InceptionV3.

B. Inception

1) Performance and Setup: The Inception V3 model was trained using TensorFlow and Keras libraries using CUDA accelerated GPU training on an Nvidia 3070 Mobile GPU with 8 gigabytes of VRAM(Video Random Access Memory). The model was trained from the "mixed 6" layer to the fully connected layer. The function used was binary cross entropy to provide probabilities between 1 and 0. A threshold was kept at 0.5 to determine whether the image showcased was classified as a defect.

2) Accuracy and Confusion Matrix : The model's accuracy against validation at the end of the training was 96%. The confusion matrix with the best result was observed at 140 steps with 35 epochs, which took around 30 mins with GPU training. This model was then used to predict 174 test images, and then its actual labels were taken from the image name to perform the comparison as shown in Fig. 7, which resulted in 94% accuracy. We also observed over-fit in the model when we trained 160 steps and 70 epochs, which gave 100% accuracy. We noted that 120 steps and 40 epochs resulted in under-fitting. The total amount of time spent trying various training and testing alone was around 4 hours.

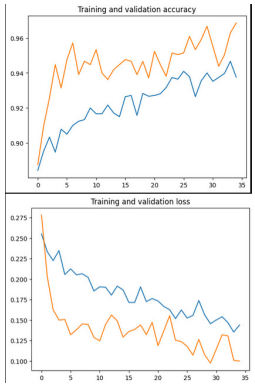

Fig. 15. Training and Validation loss.

VI. Limitations and Future Work

While our study has made significant strides in leveraging deep learning for 3D printing defect detection, some limitations warrant attention, notably the modest size of our initial data-set and the test data set we collected. The data-set comprises images of 3D printed objects from seven different 3D printers, categorizing defects such as bed not sticking, leg broken, no bottom, and no support. This limited diversity, coupled with the relatively small amount of data available, may affect the model's generalizability to a broader range of real-world 3D printing challenges. To address this limitation, future efforts should prioritize the expansion of the data-set to incorporate a more extensive and diverse set of defect scenarios encountered in the 3D printing process. By diversifying the data-set, we can compensate for the scarcity of data and empower the model to classify defects accurately in a broader range of scenarios. This approach is pivotal for contributing to the overall efficacy of the defect detection system for 3D printing. The model's capability to identify objects can be enhanced through continuous observation of patterns over additional epochs and experimentation with alternative hyper-parameters or methodologies. Rigorous exploration of these aspects may further optimize the model's performance, compensating for data limitations and refining defect detection accuracy in the 3D printing monitoring system.

VII. Conclusion

In summary, integrating deep learning, particularly the application of Convolutional Neural Networks (CNNs), within the context of 3D printing quality control represents a significant advancement in the field of 3D printing. The YOLOv8 and Inception architectures, selected for their efficacy in real-time defect detection, have demonstrated commendable performance, as evidenced by comprehensive evaluation metrics.

References

[1] "DHT11–Temperature and Humidity Sensor". [Online]. Available: https://components101.com/sensors/dht11-temperature-sensor, (accessed Dec. 10, 2023).

[2] "DHT11 Sensor and Its Working". [Online]. Available: https://www.elprocus.com/a-brief-on-dht11-sensor, (accessed Dec. 9, 2023).

[3] "Temperature and Humidity Module DHT11 Product Manual". [Online]. Available: https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf, (accessed Dec. 9, 2023).

[4] ”How to Set Up the DHT11 Humidity Sensor on an Arduino”. [Online]. Available: https://www.circuitbasics.com/wp-content/uploads/2015/12/DHT11-Pinout-for-three-pin-and-four-pin-types-2.jpg, (accessed Dec. 9, 2023).
[5] ”28BYJ-48 Stepper Motor with ULN2003 Driver and Arduino Tutorial”. [Online]. Available: https://www.makerguides.com/28byj-48-stepper-motor-arduino-tutorial/, (accessed Dec. 8, 2023).
[6] ”5V Stepper Motor 28BYJ-48 + ULN2003 Driver Test Module for Arduino,Micro Mini Electric Step Motor for PIC 51 AVR”. [Online]. Available: https://www.aliexpress.com/item/32892331606.html, (accessed Dec. 10, 2023).
[7] Raeburn, Kenneth (2005), ”Advanced Encryption Standard (AES) Encryption for Kerberos 5”, doi:10.17487/RFC3962. RFC 3962, Retrieved December 8, 2023. https://datatracker.ietf.org/doc/html/rfc3962
[8] Ruan, J. (2022). ”3D-Printer Defected Dataset”. [Online]. Available: https://www.kaggle.com/datasets/justin900429/3d-printer-defected-dataset, (accessed Dec. 10, 2023).
[9] ”Early detection of 3D printing issues”. Retrieved Dec. 10, 2023. [Online]. Available: https://www.kaggle.com/competitions/early-detection-of-3d-printing-issues
[10] Roboflow, ”Preprocess Images”. [Online]. Available: https://docs.roboflow.com/datasets/image-preprocessing, (accessed Dec. 10, 2023).
[11] Roboflow, ”Create Augmented Images”. [Online]. Available: https://docs.roboflow.com/datasets/image-augmentation, (accessed Dec. 10, 2023).
[12] Ultralytics, ”Model Training with Ultralytics YOLO”. [Online]. Available: https://docs.ultralytics.com/modes/train/, (accessed Dec. 10, 2023).
[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” arXiv.org, 2015. https://arxiv.org/abs/1512.00567, (accessed Dec. 10, 2023).
[14] Bharath Raj. ”A Simple Guide to the Versions of the Inception Network”. towardsdatascience.com. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202, (accessed Dec. 10, 2023).
[15] Qu, Zhiyu & Wang, Wenyang & Hou, Changbo & Hou, Chenfan. (2019). Radar Signal Intra-Pulse Modulation Recognition Based on Convolutional Denoising Autoencoder and Deep Convolutional Neural Network. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2935247.