

Summer Research Internship Report
on
**Unsupervised machine learning for anomaly detection in Wire-arc Additive
Manufacturing**

carried out under the mentorship of
Dr. Mukesh Chandra
Production and Industrial Engineering Department
BIT Sindri, Dhanbad



SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR SUMMER TRAINING
INTERNSHIP

Priyanshu Bist
(Roll No – 102208010)
Mechanical Engineering
Thapar Institute of Engineering and Technology
Patiala, Punjab-147004

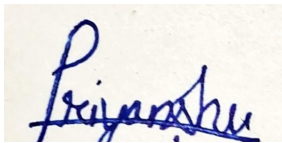


Declaration

I extend my heartfelt appreciation to **Production and Industrial Engineering Department, BIT Sindri** for granting me the opportunity for a research internship at this esteemed institute.

I am deeply grateful to **my mentor, Dr. Mukesh Chandra**, whose unwavering guidance and encouragement introduced me to the fascinating field of research. His mentorship has been instrumental in shaping my understanding and approach.

I hereby declare that this written submission reflects my own ideas expressed in **my own words**. Where I have included the ideas and words of others, I have appropriately cited and referenced the original sources. I affirm that I have adhered to all principles of academic honesty and integrity. **There has been no misrepresentation, fabrication, or falsification of any idea, data, fact, or source in my submission..** I acknowledge that any violation of the above principles may result in disciplinary action by the Institute. Furthermore, failure to properly cite sources or obtain necessary permissions may lead to penal action from the concerned parties.



Priyanshu Bist

Date: 15-09-2024

Overview

Wire-arc additive manufacturing (WAAM) has emerged as the most economical metal additive manufacturing process for large-volume and complex structures in the aerospace and construction industries. The quality of deposited material layer by layer is sacrificed due to the unavailability of an advanced monitoring and control system. This report uses **unsupervised machine learning (UML)** to detect anomalies in the deposited layers. The melt pool images of the WAAM process were collected by conducting experiments at different process parameters.

Different UML techniques, such as **K-means clustering, Isolation Forests, and Principal Component Analysis (PCA)**, are used to understand the underlying pattern in melt pool images of WAAM.

Image preprocessing techniques, including **contrast adjustment, noise reduction, and edge detection**, were applied to enhance feature extraction. The **EfficientNetB0** model was used for feature extraction, followed by **PCA** for dimensionality reduction.

Performance evaluation utilized metrics such as **Silhouette Score, Calinski-Harabasz Index, Davies-Bouldin Index, Precision, Recall, and F1 scores** along with **visualizations graphs** such as **AUC/ROC and Precision-Recall Curves**.

Some key findings include:

- **Case 1** demonstrated improved performance when both training and testing datasets were labelled, with an AUC-ROC score of **0.92** compared to **0.62** for labelled testing only.
- **Cases 2, 3, and 6** showed **varying clustering performance** as the number of classes increased, with generally **consistent** results for **3-class scenarios**.
- **Cases 4 and 5**, focusing on binary classification, achieved moderate performance with AUC-ROC scores of **0.55** and **0.56**, respectively.

Across all cases, the application of **SMOTE** for dataset balancing and outlier detection using **Isolation Forests** contributed to improved model performance.

Contents

S. No.	Topic
1.	Declaration
2.	Overview
3.	Datasets Details
4.	Initial Algorithm Planned
5.	Progress throughout Program
6.	Algorithms covering all Cases
7.	Research Results (link)
8.	Conclusion
9.	References (link)

1 Data Details

Case	Dataset details	Number of Target clusters	Objectives	Remark
Case_I	All regular and regular bead images are merged together	2	To classify the regular and irregular bead using melt pool images. Irregular bead images are anomalies here.	Labelled testing is provided
Case_2	Regular bead with anomaly	More than 2	To detect anomalies in regularly deposited beads, anomalies may be defects present in the deposited bead, such as porosity, cracks, etc., and different numbers of clusters could be found. It opens possibility for quality control in WAAM process.	Labelled testing dataset in not possible,
Case_3	Irregular bead with anomaly	More than 2	The formation of irregular beads is itself an anomaly in wire-arc additive manufacturing (WAAM). However, different cluster formations using melt pool images of irregular beads can be further analysed for a better understanding of the dataset. It opens possibility for quality control in WAAM process.	Labelled testing dataset in not possible,
Case_4	Regular bead with spatter	2	Spatter images are anomalies here.	Labelled testing is provided
Case-5	Regular bead with extinct arc	2	Extinct arc images are anomalies here.	Labelled testing is provided
Case-6	Regular bead with variation in bead surface roughness	2	Anomaly in regularly deposited bead might occur due the its surface roughness and waviness. Deposited beads with greater surface roughness and waviness are anomalies here.	Labelled testing is provided. In this case, the test dataset did not merge because the file name of the images is the same. Change the file of images to one folder; then, the dataset will be split.

2 Intital Algoritithm Planned

1. **Data Collection:** Gather a dataset of images.
2. **Data Preprocessing:**
 - Resize images to a consistent size.
 - Normalize pixel values (e.g., scale to $[0, 1]$ or $[-1, 1]$).
 - Data augmentation (optional): apply random transformations to increase diversity.
3. **Feature Extraction:**
 - Use techniques like convolutional neural networks (CNNs) or traditional computer vision methods (e.g., SIFT, HOG) to extract features from images.
4. **Clustering:**
 - Apply clustering algorithms (e.g., K-Means, Hierarchical Clustering, DBSCAN) to group similar images based on their features.
5. **Dimensionality Reduction (optional):**
 - Use techniques like PCA, t-SNE, or Autoencoders to reduce the feature space dimensionality for visualization or improved clustering.
6. **Anomaly Detection (optional):**
 - Identify outlier images that don't fit the clustering structure.
7. **Evaluation:**
 - Assess the quality of clustering using metrics like Silhouette Score, Calinski-Harabasz Index, or Davies-Bouldin Index.
8. **Visualization (optional):**
 - Use dimensionality reduction techniques to visualize the clustering structure.

Popular Unsupervised Learning Algorithms for Images

1. K-Means
2. Hierarchical Clustering
3. DBSCAN
4. Autoencoders
5. t-SNE (t-Distributed Stochastic Neighbor Embedding)
6. U-Net (for image segmentation)

3 Progress throughout entire Research Program

Across the six cases, we experimented with various clustering techniques, including DBSCAN, K-Means, and Agglomerative Clustering. K-Means consistently provided satisfactory results, so we applied it across all cases. We also **addressed dataset imbalance using SMOTE and applied Isolation Forest for outlier detection**. In some instances, especially when the dataset was imbalanced or feature detection was insufficient, Isolation Forest results were unusable (**NaN values**). By using SMOTE and ensuring proper feature detection, we improved classification accuracy. In Case 1, the model performed significantly better when both the **training and testing datasets were labeled**. This was contrasted with Cases 4 and 5, where labeling both the training and testing datasets was essential for generating valid results. **Image processing techniques** were crucial in Cases 4 and 5 due to the presence of splatters or dull features in the melt pool images, which initially hindered the model's ability to detect anomalies effectively. Enhancing image quality through processing improved performance.

For Cases 2, 3, and 6, we focused on evaluating the model's consistency and effectiveness by varying the number of clusters (2, 3, 4, 5) and analyzing the evaluation scores, such as Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index. This allowed us to

assess how well the model performed under different clustering scenarios, ensuring robustness and reliability.

A major focus was understanding the melt pool images before implementing image processing steps. Significant debugging was done to address dataset issues, and all cases were finally evaluated using **K-Means** and other evaluation tools to **refine and secure** the results.

4 Algorithms covering all Cases

4.1 Case 1 (when testing and training datasets were labeled): Algorithm

1. Importing Libraries
2. Define Dataset Paths
3. Image Loading and Preprocessing
 - (a) Load images from the specified folder.
 - (b) Resize images to (224, 224).
 - (c) Convert images to arrays and preprocess for ResNet50.
 - (d) Assign labels based on folder name (class 0 or class 1).
4. Load and Concatenate Datasets
5. Feature Extraction
 - Load ResNet50 model (excluding top layers, using average pooling).
 - Extract features for both training and testing images using ResNet50.
6. Dimensionality Reduction
7. Clustering

- Define the number of clusters (2).
- Apply KMeans clustering to the PCA-transformed training features.
- Predict cluster labels for training data.

8. Clustering Evaluation

- Define `evaluate_clustering` function:
 - (a) Compute silhouette score.
 - (b) Compute Calinski-Harabasz index.
 - (c) Compute Davies-Bouldin index.
- Evaluate KMeans clustering performance on training data.

9. Anomaly Detection

- Apply Isolation Forest to detect anomalies in the training features.
- Filter out anomalies from training features and labels.

10. Clustering on Cleaned Data

- Apply KMeans clustering to the cleaned training features.
- Evaluate clustering performance on cleaned data.

11. Performance Metrics for KMeans

- Compute precision, recall, and F1-score for KMeans clustering on cleaned data.

12. SMOTE for Balancing

- Apply SMOTE to balance the dataset.
- Fit KMeans clustering to the SMOTE-resampled training features.

13. Clustering Evaluation on SMOTE Data

- Evaluate KMeans clustering performance on SMOTE-resampled data.

14. 3D Clustering Visualization
15. Confusion Matrix Plotting
16. ROC Curve Plotting
17. Print Evaluation Metrics

4.2 Case 1 (when only datasets were labeled): Algorithm

1. Importing Libraries
2. Define Dataset Paths
3. Image Processing Function (same as Case 4 Algorithm)
4. Loading and Preprocess Images
5. Checking the Distribution of Classes in the Training Set
6. Only Proceed with SMOTE if There is More than One Class
7. Feature Extraction Using EfficientNetB0
8. Apply PCA
9. Apply K-Means Clustering with k-means++ Initialization
10. Evaluate Clustering Performance
11. Additional Evaluation Indexes
12. Print Evaluation Metrics and Classes Distribution
13. 2D PCA Visualization of K-Means Clusters
14. 3D PCA Visualization of K-Means Clusters
15. Visualization of True Labels in 2D PCA

16. Confusion Matrix Heatmap
17. ROC Curve
18. Precision-Recall Curve

4.3 Case 2, Case 3, and Case 6: Algorithm

1. Importing Libraries
2. Define Data Folder
3. Image Loading and Preprocessing
4. Feature Extraction
5. Dimensionality Reduction
6. VAE Encoding (Placeholder)
7. Clustering
8. Clustering Evaluation
9. SMOTE for Balancing
10. Anomaly Detection
11. 3D Clustering Visualization
12. ROC Curve Plotting
13. Confusion Matrix Plotting
14. Classification Report
15. Print Completion Message

4.4 Case 4 Algorithm Steps (when training dataset is labeled) with UNLABELED code isn't working

1. Importing Libraries
2. Defining KMeansWrapper Class
3. Paths to Datasets and Storage
4. Image Processing Function Made
 - (a) Contrast adjustment using CLAHE and Gamma Correction
 - (b) Convert the image back to 8-bit
 - (c) Advanced noise reduction using Non-Local Means Denoising
 - (d) Apply Canny edge detection
 - (e) Thresholding using Otsu's method
 - (f) Morphological operations to remove noise
 - (g) Masking
 - (h) Normalize the image
 - (i) Convert back to uint8
 - (j) Stack to 3 channels for compatibility with pre-trained models
5. Loading and Preprocessing Images
6. Applying SMOTE for Balancing the Dataset
7. Feature Extraction Using EfficientNetB0
8. Applying PCA for Dimensionality Reduction
9. Outlier Detection Using Isolation Forest
10. Keeping Only Inliers

11. Defining Parameter Grid
12. Setting Up Grid Search
13. Applying the Best K-Means Model
14. Evaluating Clustering Performance
15. Additional Evaluation Indexes
16. ROC and Precision-Recall Curves
17. Printing Evaluation Metrics
18. Other Visualizations
 - (a) Confusion Matrix Heatmap
 - (b) PCA Components Plot
 - (c) ROC Curve
 - (d) Precision-Recall Curve
 - (e) Clustering Results
 - (f) Anomaly Detection Plot

4.5 Case 5 Algorithm Steps (when training dataset is labeled) with UNLABELED code isn't working

1. Importing Libraries
2. Paths to Datasets
3. Image Processing Function
 - (a) Convert to Grayscale: Read and convert the image to grayscale.
 - (b) Resize Image: Resize the image to a standard size (128x128).

- (c) Contrast Adjustment: Apply CLAHE (Contrast Limited Adaptive Histogram Equalization).
- (d) Convert to 8-bit: Convert the image back to 8-bit.
- (e) Gaussian Blur: Apply Gaussian Blur for noise reduction.
- (f) Edge Detection: Use Canny edge detection to highlight edges.
- (g) Thresholding: Apply Otsu's method for binary thresholding.
- (h) Morphological Operations: Remove small objects and apply morphological closing.
- (i) Masking: Apply the binary mask to the original image.
- (j) Normalization: Rescale intensity to normalize the image.
- (k) Convert Back to uint8: Convert the processed image back to uint8 format.
- (l) Stack Channels: Stack the processed image to 3 channels for compatibility with pre-trained models.

4. Image Augmentation Function

5. Loading and Preprocessing Images

6. Augmenting the Data

7. SMOTE for Dataset Balancing

8. Feature Extraction Using EfficientNetB0

9. PCA for Dimensionality Reduction

10. Clustering Using K-Means

11. Evaluation Metrics Calculation

- (a) Confusion Matrix: Compare predicted labels against true labels.
- (b) Accuracy: Compute balanced accuracy.

- (c) Precision: Compute precision score.
- (d) Recall: Compute recall score.
- (e) Specificity: Calculate specificity.
- (f) False Positive Rate (FPR): Compute FPR.
- (g) F1 Score: Calculate F1 score.
- (h) MCC: Compute Matthews Correlation Coefficient.
- (i) Silhouette Score: Calculate silhouette score.
- (j) Calinski-Harabasz Index: Calculate Calinski-Harabasz index.
- (k) Davies-Bouldin Index: Calculate Davies-Bouldin index.

12. Visualizations

- (a) 2D PCA Cluster Visualization: Scatter plot of the first two PCA components colored by cluster labels.
- (b) 3D PCA Cluster Visualization: 3D scatter plot of the first three PCA components colored by cluster labels.
- (c) Confusion Matrix Heatmap: Heatmap visualization of the confusion matrix.
- (d) ROC Curve: Plot the Receiver Operating Characteristic curve.
- (e) Precision-Recall Curve: Plot the Precision-Recall curve.

13. Printing Evaluation Metrics

5 Research Results

For detailed results of the research, please refer to the following link:

Research Results

6 Conclusion

In our work, significant progress has been achieved in developing and refining algorithms for all cases, especially Cases 4 and 5 where image processing was needed.

Our efforts have been **focused on improving** the performance metrics of our algorithms. By implementing advanced image processing techniques, optimizing feature extraction, and utilizing sophisticated clustering and anomaly detection methods, we have achieved notable improvements in evaluation scores. This includes enhanced precision, recall, and F1-scores, reflecting better accuracy in anomaly detection and clustering performance.

Despite these advancements, there is **still room for improvement**. We continue to refine our algorithms to further enhance their effectiveness and accuracy. Our ongoing efforts aim to address remaining challenges and achieve even better results.

7 References

All datasets and research papers can be accessed through the following OneDrive link:

Datasets and Research Papers