MODELLING OF MANUFACTURING PROCESSES USING MACHINE LEARNING

Project report submitted in partial fulfilment of the

requirements for the degree of

Bachelor of Technology

in

Mechanical Engineering

by

Nidhi Singh
(Roll No. 210103127)

Saurabh Kumar
(Roll No. 210103099)

Under the supervision of

Dr. Sukhomay Pal

to

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI – 781039, INDIA

[Jul.2024 - Nov. 2024]

Acknowledgement

We owe our heartfelt gratitude and deepest regard to our supervisor, Dr. Sukhomay Pal, Department of Mechanical Engineering, IIT Guwahati, India, for his invaluable mentorship, constant guidance, and unwavering support throughout the project. His profound knowledge, vision, and enthusiasm have been a constant source of inspiration and motivation for us. This project would not have been possible without his insightful feedback and direction.

We extend our sincere gratitude to our project guides, Mr. Shekhar Verma, MTech senior, and Mr. Vivekanand, Ph.D. senior, for their consistent assistance, encouragement, and valuable inputs throughout the course of this project.

We express our sincere appreciation to the Head of the Department of Mechanical Engineering, IIT Guwahati, for providing us with an excellent research environment and the necessary infrastructure that made this work possible. The access to advanced research facilities and resources was crucial in executing our project successfully.

Our gratitude extends to the entire Department of Mechanical Engineering, IIT Guwahati, including faculty members, technical and non-technical staff, and research scholars, for their continued support and encouragement. We are especially thankful for the facilities provided, which allowed us to carry out our research seamlessly.

Abstract

Advancements in material characterization and machine learning have paved the way for innovative approaches to microstructure analysis. This study focuses on developing robust machine learning models for grain boundary detection in microstructure images, leveraging Convolutional Neural Networks (CNNs) and transfer learning techniques. A dataset comprising 480 microstructure images with meticulously hand-annotated masks was used to train and evaluate various deep learning architectures, including a custom CNN, Xception, and U-Net models. Among these, the U-Net architecture with transfer learning demonstrated the highest accuracy (87%) in predicting grain boundaries, owing to its encoder-decoder design and skip connections that preserved fine spatial details.

The methodology included data preprocessing, manual annotation, and segmentation, followed by iterative model training and optimization. High-resolution images of 316L stainless steel samples, was obtained from the journal that was prepared through precision techniques like grinding, polishing, and etching, were utilized for dataset creation. However, the next step is training on self-made dataset and other enhancements. Quantitative metrics such as Dice coefficient and accuracy were employed to evaluate model performance, with the U-Net model achieving superior boundary delineation and segmentation outcomes.

This research establishes a foundation for integrating advanced deep learning methodologies into material science, highlighting the potential for improved grain boundary prediction and quantitative microstructure analysis. Future work aims to extend these models to diverse materials and explore contour-based methods for enhanced precision in boundary detection.

Certificate

This is to certify that the work contained in this project report entitled **"Modelling of Manufacturing Processes Using Machine Learning"** submitted by **Nidhi Singh (210103127)** and **Saurabh Kumar (210103099)** to the Indian Institute of Technology Guwahati towards the partial requirement of **Bachelor of Technology** in **Mechanical Engineering** is a Bonafide work carried out by them under my supervision and that it has not been submitted elsewhere for the award of any degree.

Dr. Sukhomay Pal
Professor
Department of Mechanical Engineering
Indian Institute of Technology Guwahati
Nov. 2024

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Nidhi Singh
Roll No. 210103127

Saurabh Kumar
Roll No. 210103099

Contents

Nomenclature

Abbreviations

Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
BGD	Batch Gradient Descent
CNN	Convolutional Neural Network
DL	Deep Learning
FP	False Positive
FN	False Negative
MAE	Mean absolute error
ML	Machine Learning
MAPE	Mean Absolute Percentage Error
MSE	Mean Square Error
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
TN	True Negative
TP	True Positive
UTM	Ultimate Tensile Strength

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, the adoption of machine learning (ML) has surged across a wide range of applications, including multimedia retrieval, spam detection, text mining, and image categorization. Deep Learning (DL), a specialized subset of ML, has led this transformation by building upon traditional neural networks to create complex, multi-layered models through advanced transformation techniques and graph-based technologies. The capabilities of DL have expanded rapidly, achieving exceptional results in areas such as Natural Language Processing (NLP), audio and speech analysis, and visual data processing.

A critical factor in the success of any ML algorithm is the quality of the input data representation. Research consistently shows that well-crafted data representations substantially improve model performance. Feature engineering, aimed at producing meaningful and relevant features from raw data, has therefore long been central to ML research, requiring considerable human expertise and customization for specific domains. For instance, computer vision research has developed and refined various types of feature representations. DL models, however, excel at automating this feature extraction process: initial layers capture fundamental, low-level features, while deeper layers progressively identify more complex, high-level information. This layered architecture minimizes the need for manual feature engineering, positioning DL as a powerful tool in domains where feature extraction is particularly challenging.

1.2 Motivation of the Study

Materials by Design involves engineering materials from the atomic to the macroscopic scale to achieve specific performance targets. This forward-thinking approach has the potential to revolutionize material innovation, particularly in fields such as renewable energy, energy storage, sustainable materials, and critical infrastructure. Rapid advancements in this domain have been driven by the growing use of computational tools, expansive materials databases, refined experimental techniques, and the integration of artificial intelligence and machine learning into the materials discovery process. However, despite notable progress, the field is still relatively young and faces several key challenges.

One of the primary challenges in Materials by Design is identifying microstructural features that are essential for developing accurate processing-structure-properties (PSP) models. While microstructural analysis has a long history, integrating this valuable information into this domain has been challenging, largely because microstructural descriptions have traditionally been qualitative. To build precise PSP models, it is essential to extract quantitative, statistically meaningful data from microstructural analyses. Although various methods exist for characterizing and reconstructing microstructures, many result in substantial information loss, limiting their suitability for the precision required in Materials by Design.

1.3 Objective of the present study

The primary objective of this study is to develop CNN models capable of predicting and identifying microstructural properties of materials directly from image data, thereby minimizing human intervention.

The specific objectives to achieve this are as follows:

1. Develop a machine learning model to observe and analyze microstructures.
2. Employ CNN models to accurately identify grain boundaries and grains within the input images.
3. Implement multiple model architectures to enhance accuracy in predicting grain properties.
4. Evaluate the influence of various hyperparameters on the prediction accuracy of the models.

1.4 Organization of the Thesis

The thesis has been organized keeping in mind the overall path followed to achieve the final aim i.e. to characterise the wake when a square body is transformed to a triangular body. **Chapter 1** primarily highlights the motivation behind carrying out the present work. It further offers a background on the bluff bodies and its interaction with the flow field. **Chapter 2** covers a literature review on major research topics in the field bluff body flow dynamics. **Chapter 3** covers the methodology of the work and validation with the available literatures. **Chapter 4** consists of the results and its explanation on account of this transformation. In the concluding chapter i.e. in **Chapter 5**, the key findings and some future scopes are highlighted.

1.5 Review of BTP Phase I and Phase II

In the previous semester, we built foundational knowledge in deep learning through Andrew Ng's specialization, focusing on applying deep learning algorithms to monitor manufacturing processes. This involved working with datasets containing features like machining time, surface roughness, MRR, and feed rate, while also experimenting with Convolutional Neural Networks (CNNs) for image analysis to optimize these processes.

The microstructure classification model developed in Phase 1 and Phase 2 shows promising potential but requires refinement. Despite achieving higher accuracy, small variances in certain variables need further attention for better image moderation and expansion. With continued improvements and collaboration with domain experts, the model can be a valuable tool for material characterization and manufacturing optimization, with further research needed to unlock its full potential.

CHAPTER 2

LITERATURE REVIEW

2.1 Deep Learning and Microstructures

The prediction of a product's microstructure under specific processing conditions has gained significant attention in advanced manufacturing and material science due to the critical influence of microstructure on material properties. Traditionally, this knowledge was obtained through experimental trial-and-error methods, which are time-consuming and resource-intensive. However, recent advancements in physics-based models, which predict microstructures based on established laws, have been developed. Despite their potential, these models often come with high processing costs and are unreliable for real-time forecasting in modern production environments.

In contrast, machine learning (ML) approaches, particularly data-driven techniques, offer effective alternatives for simulating material microstructures. These ML algorithms can generate simulation results almost instantaneously after training, without requiring detailed knowledge of the underlying physical principles. Neural networks, with their various architectures, have been employed to address several challenges in microstructure analysis, including the identification of microstructural fingerprints, detection of weld joint flaws, grain analysis, volume phase maintenance, and the prediction of phase constituents in steel during thermomechanical processing and cooling. These methods provide a promising direction for enhancing the prediction and optimization of material properties.

2.2 Convolutional Neural Network Background

Convolutional Neural Networks (CNNs) have become pivotal in engineering research due to their capacity to model complex relationships, especially in computer vision. A key advantage of CNNs is their ability to automatically identify significant features in images, minimizing the need for manual feature selection. This functionality has made CNNs exceptionally popular for image-based tasks

across fields such as computer vision, robotics, and autonomous vehicles, though their application in microstructure analysis remains relatively new.

CNNs excel at image classification, where each image is assigned one or more labels corresponding to distinct classes. Leveraging their properties of translation invariance and weight sharing, CNNs can efficiently handle various visual data. For instance, CNNs have been applied to predict material properties from images of material topologies, offering a direct approach to analyzing composite structures—where conventional fully connected networks fall short due to the complexity of composite material data that resists simple vectorization.

Widely used in supervised learning, CNNs are predominantly employed for classification, supporting tasks from handwriting and spam detection to real-time object recognition. With the "SoftMax" activation function, CNNs can predict classes across thousands of categories, accommodating binary and multi-class outputs alike. Moreover, CNNs can be adapted for regression tasks, where decimal values are predicted instead of discrete labels, enabling applications such as facial landmark localization, head and human pose estimation, and housing price predictions. This versatility in handling both categorical and continuous data makes CNNs invaluable across diverse domains.

2.3 Convolutional Neural Network Architecture

The Convolutional Neural Network (CNN) architecture is built upon three primary layers, each serving a unique function to process and analyse image data effectively. Below is a refined description of each layer type in a CNN.

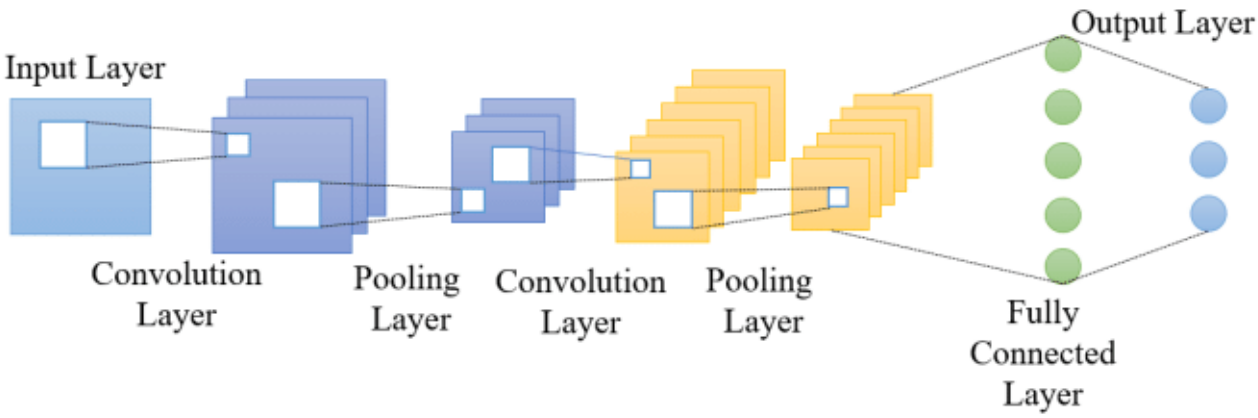


Fig. 2.3 CNN Architecture

(A) Convolutional Layer

The convolutional layer is the core feature extraction component in CNNs. This layer applies filters, typically of size M×M, to the input image, performing a mathematical operation called convolution. Each filter, also referred to as a kernel, slides across the image, interacting with specific regions of pixels. This operation generates a feature map that captures essential details of the image, such as edges, textures, and patterns. By learning multiple filters, CNNs can extract and identify various characteristics of an image, making the model highly effective in recognizing complex structures.

(B) Pooling Layer

The pooling layer follows the convolutional layer and is responsible for reducing the dimensions of the feature maps, thereby minimizing computational complexity while retaining important information. Pooling is typically performed using max pooling (selecting the maximum pixel value in each region) or average pooling (calculating the average pixel value). This layer helps to decrease the model's sensitivity to slight shifts and distortions in the image, further enhancing the translation invariance properties of CNNs.

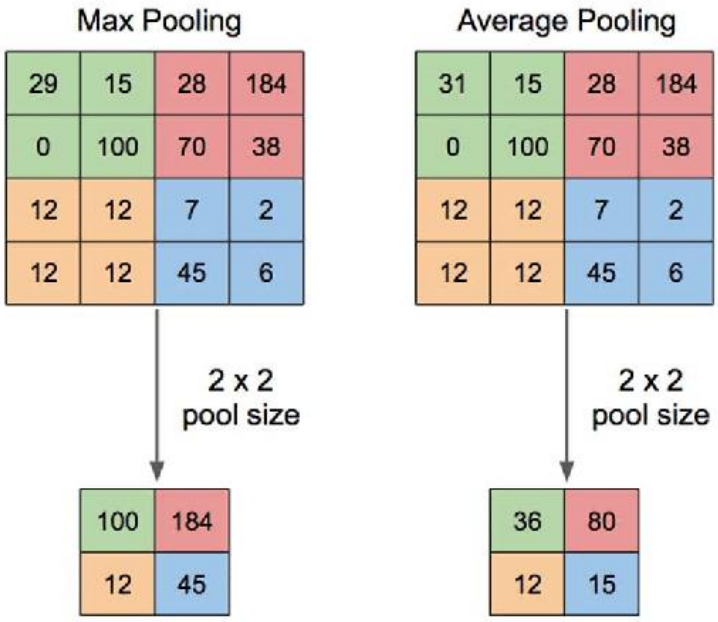


Fig. 2.4 Pooling Layer

(C) Fully Connected Layer

The fully connected (FC) layer usually appears at the end of a CNN structure. In this layer, each neuron connects to every neuron in the preceding layer, creating a dense network that combines the extracted features to make predictions. The FC layer receives inputs from the pooled feature maps, which are flattened into a vector. This layer integrates the spatially reduced features, enabling the CNN to classify images based on the learned representations. The final output layer is typically part of the FC layer and produces the predicted class or value.

This layered architecture, with convolutional, pooling, and fully connected layers, allows CNNs to process images with high accuracy and efficiency across diverse domains, such as object detection, material property prediction, and microstructure classification.

(D)Activation Function

The activation function is typically placed following a fully connected layer to introduce nonlinearity to the network. Some commonly employed activation functions are the Sigmoid function, ReLU, and tanh.

(i) Sigmoid function: This activation function restricts the input and output values to the range of 0 to 1. The sigmoid function has a characteristic "S" shape and is expressed mathematically in Equation 2.1.

$$f_{\text{sigmoid}} = \frac{1}{1+e^{-x}} \tag{2.1}$$

(ii) Tanh function: This function functions in a similar manner to the sigmoid function, taking real numbers as input but constraining the output within a specific range of -1 to 1. Equation 2.2 represents its mathematical form.

$$f_{\text{tanh}} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.2}$$

(iii) ReLU function: In the context of Convolutional Neural Networks (CNNs), this function is extensively used and is considered the most popular choice. It transforms the integer inputs into positive values. The major advantage of the Rectified Linear Unit (ReLU) over other activation functions is its computational efficiency, making it computationally lighter. Equation 2.3 presents its mathematical representation.

$$f_{\text{ReLU}} = max(0, x) \tag{2.3}$$

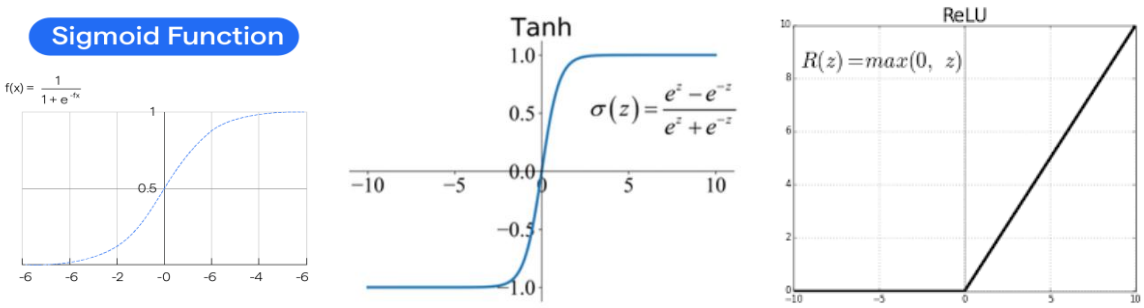


Fig. 2.5 (A) Sigmoid Function (B) Tanh Function (C) ReLU Function

2.4 Operations performed in CNN

(a) Convolutional Operation

In neural networks, the convolution operation involves the interaction between two tensors, which serve as inputs to produce an output tensor. This operation is denoted by the "*" symbol. Here, the input microstructure image is represented as X, and the filter is represented by f. Thus, the convolution operation can be expressed as:

$$Z = X \times f \tag{2.4}$$

If the input image has dimensions n×n and a filter of size f×f is applied, the resulting output image will have dimensions of $(n - f + 1) \times (n - f + 1)$.

(b) Max Pooling

Max pooling is a widely used operation in convolutional neural networks (CNNs) for feature extraction and down-sampling. It is a type of pooling or subsampling that reduces the spatial dimensions (width and height) of an input volume, such as an image or feature map, while preserving the most significant information.

(c) Flattening Operation

The fully connected layer, which produces the final output, receives features extracted by a specific convolutional layer from the data. In a Convolutional Neural Network, this fully connected layer generally functions like a traditional Neural Network. Notably, the fully connected layer requires a one-dimensional input, whereas the convolutional layer provides a two-dimensional matrix as its output.

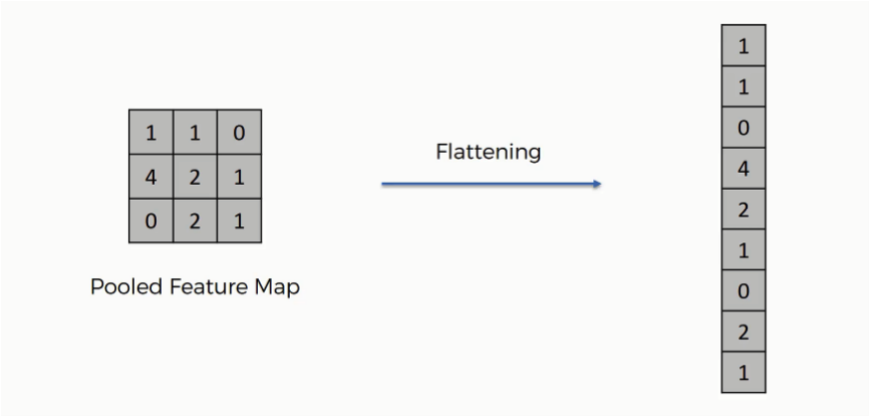


Fig. 2.6 Flattening Operation

(d) Dense Layers Operation

Once the two-dimensional output from the convolutional layers is flattened into a one-dimensional array, each value is treated as a distinct feature. In the fully connected layer, the data first undergoes a linear transformation, where values are weighted and summed with a bias term. This is followed by a non-linear transformation (activation function), allowing the network to capture complex patterns and produce the final output, such as class probabilities.

2.5 Metrics to measure accuracy of CNN

I. Mean Squared Error (MSE): Mean Squared Error is a metric for assessing the accuracy of a regression model. In CNNs, MSE measures the average squared difference between predicted values and actual target values. Mathematically, it is represented as:

$$MSE = \frac{1}{n}\sum(y_i - y_i')^2 \quad (2.5)$$

Where,

n is the number of data points,

y_i is the actual value for the i^{th} data point,

y_i' is the predicted value, and $\sum$ represents the summation across all data points.

II. Mean Absolute Error (MAE): The Mean Absolute Error is a metric used in regression tasks, including CNN evaluations. It calculates the average of the absolute differences between predicted and actual target values. Mathematically, it is expressed as:

$$MAE = \frac{1}{n}\sum |y_i - y_i'| \quad (2.6)$$

where the variables are defined as in MSE. MAE measures the size of errors without regard to their direction.

III. Mean Absolute Percentage Error (MAPE): MAPE is a commonly used metric for evaluating model performance, particularly in forecasting and regression. It calculates the average absolute percentage difference between predicted and actual values, defined mathematically as:

$$\text{MAPE} = \frac{1}{n}\sum \left| \frac{(yi - yi')}{yi} \right| \times 100 \tag{2.7}$$

where the variables have the same meanings as in the MSE definition. MAPE offers insight into the error size relative to the actual values, making it helpful for understanding prediction accuracy in relation to data scale. However, MAPE can be unreliable when target values are near zero, potentially causing undefined or excessively large results due to division by zero.

IV. F1 Score/Dice Coefficient: The F1 Score is a key metric for evaluating classification models, often used in CNNs for tasks like image classification. It combines precision and recall, making it especially valuable for imbalanced datasets. The F1 Score is given by:

$$\text{F1}_{\text{score}} = \frac{2(\text{Precision} * \text{Recall})}{(Precision + Recall)} \tag{2.8}$$

where:
Precision is the ratio of true positive predictions to all positive predictions:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.9}$$

Recall (or sensitivity) is the ratio of true positive predictions to actual positive instances:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.10}$$

The F1 Score combines precision and recall into a single measure, making it useful for both binary and multiclass classification tasks, particularly when class distributions are uneven.

V. Pixel Accuracy: Pixel accuracy calculates the ratio of correctly classified pixels to the total number of pixels. For K + 1 classes (where K represents foreground classes and the background class), pixel accuracy is defined as:

$$PA = \frac{\sum_{i=0}^{K} pii}{\sum_{i=0}^{K} \sum_{j=0}^{K} pij} \quad (2.11)$$

where p_{ij} denotes the number of pixels from class i that are predicted as class j. These metrics are essential for evaluating model quality in CNN-based tasks, providing insight into model performance across machine learning and deep learning applications.

2.6 Optimizer selection for minimising loss

Gradient-based Learning Algorithm or Gradient Descent: This algorithm iteratively adjusts the network's parameters during training to reduce the training error. It involves calculating the gradient (slope) of the objective function by differentiating network parameters with respect to the first-order derivative. Parameters are then updated in the opposite direction of the gradient to minimize error. This adjustment process occurs through backpropagation, which propagates the gradient from each neuron to those in the previous layer. The process is mathematically expressed as:

$$w_{ij}^{t} = w_{ij}^{t-1} - \Delta w_{ij}^{t}, \quad \Delta w_{ij}^{t} = \eta \frac{\partial E}{\partial wij} \quad (2.12)$$

Here, w_{ij}^{t} is the weight at the current epoch, w_{ij}^{t-1} is the weight from the previous epoch, E represents the prediction error, and η denotes the learning rate. Various alternative learning algorithms to gradient-based methods are also commonly available.

I. Batch Gradient Descent (BGD): In Batch Gradient Descent, the network parameters are updated only once per training cycle, using the entire training dataset. Specifically, it calculates the gradient across the full dataset and uses this to adjust the parameters. This method enables the CNN model to converge more quickly and provides a stable gradient for smaller datasets. However, BGD is resource-intensive, as parameters are updated only once per cycle. With larger datasets, convergence can take longer, and in nonconvex situations, the algorithm may settle at a local optimum.

II. Stochastic Gradient Descent (SGD): In Stochastic Gradient Descent, parameters are updated after each individual training sample is processed. For each epoch, it’s recommended to randomly shuffle the training samples before training begins. Compared to Batch Gradient Descent (BGD), SGD is faster and uses less memory, especially with large datasets. However, the frequent updates cause a noisier progression toward the solution, resulting in less stable convergence.

III. Mini-batch Gradient Descent: In Mini-batch Gradient Descent, the training data is divided into several mini-batches, each containing a small, randomly selected subset of samples. The gradient is calculated for each mini-batch, followed by parameter updates. This approach combines the benefits of Batch Gradient Descent (BGD) and Stochastic Gradient Descent (SGD), providing stable convergence, better computational efficiency, and optimized memory usage. The next section covers various optimization methods used with gradient-based algorithms, particularly in the context of SGD, to enhance CNN training.

IV. Momentum: In neural networks, the momentum method is used within the objective function to enhance both accuracy and training speed. By incorporating the gradient from the previous training step, adjusted by a momentum factor, it helps the model avoid becoming trapped in a global minimum and instead converge toward a local minimum. This addresses a common issue in gradient-based methods, where local minima are encountered in non-convex solution spaces. The momentum method is mathematically represented as:

$$\Delta w_{ij}^{t} = \eta \frac{\partial E}{\partial w_{ij}} + \alpha(\Delta w_{ij}^{t-1}) \tag{2.13}$$

where η is the learning rate, and α is the momentum factor, which controls the influence of the previous weight change w_{ij}^{t-1}. A smaller α reduces the chance of escaping local minima, while a larger α leads to faster convergence. However, a high learning rate combined with large momentum may cause the model to overshoot the global minimum. Choosing an optimal momentum factor smooths weight updates when gradient directions vary during training.

V. Adaptive Moment Estimation (Adam): Adam is a popular optimization algorithm specifically designed for deep learning, known for its memory efficiency and reduced computational cost. Adam dynamically adjusts the learning rate for each parameter based on a combination of RMSprop and momentum. Similar to RMSprop, it averages squared gradients over time to adapt the learning rate, while also incorporating momentum for further optimization.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter describes the methodology adopted for predicting grain boundaries in microstructure images. Due to the complex nature of microstructural boundaries, high-resolution segmentation models are essential for accurate boundary delineation. We explored three deep learning-based approaches: a custom Convolutional Neural Network (CNN) developed from scratch, transfer learning with the Xception model, and transfer learning with the U-Net architecture. This chapter covers the dataset creation, preprocessing, model architectures, training processes, and evaluation metrics in detail.

3.2 Training Dataset

The initial dataset utilized in this study was sourced from the research article titled *"Grain and Grain Boundary Segmentation Using Machine Learning with Real and Generated Datasets" Ref. [8]&[9]*. This dataset provided a foundation for developing and refining our boundary detection models. Comprising 480 real microstructure images paired with 480 hand-annotated masks, it offers a diverse set of grain structures essential for model generalization.

The dataset contains microstructure images that display varied grain patterns. These patterns often appear blurred or noisy, making the task of boundary detection challenging. To achieve precise boundary labelling, each image underwent careful manual annotation. This process ensured that the dataset captured the complex grain boundaries accurately. Below are the key preparation steps for this dataset:

1. **Data Collection**: The dataset includes microstructure images showcasing a range of grain shapes and boundary types, promoting model robustness by enhancing its capacity to generalize across different grain characteristics.

2. **Manual Annotation**: Each image's grain boundaries were manually traced to generate precise boundary masks. Annotators meticulously marked boundary pixels at a fine-grained, pixel-level resolution. These annotations created binary masks, with boundary pixels marked as 1 (white) and all other regions set to 0 (black).

3. **Image Preprocessing**:
    - **Resizing**: All images and their masks were resized to a standardized resolution, matching the input size requirements of the deep learning models.
    - **Normalization**: Image pixel values were normalized to a [0, 1] scale, ensuring consistent data scaling across the dataset. This process contributed to improved model stability during training.
    - **Data Splitting**: The dataset was divided into training, validation, and test subsets. Each subset was curated to include images with diverse boundary characteristics, facilitating the model's ability to generalize well.

The resulting dataset contains high-quality input images paired with binary boundary masks, enabling the models to perform a precise pixel-level classification task. This dataset, with its meticulous annotations and consistent preprocessing, forms a reliable basis for advancing grain boundary segmentation through machine learning.

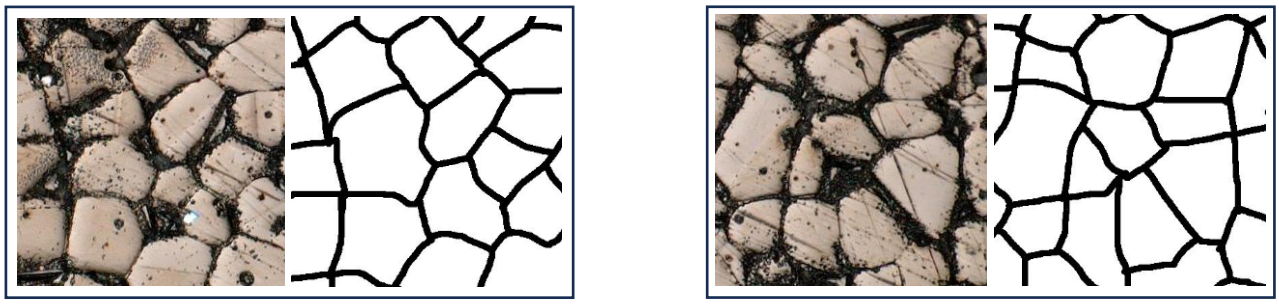


Fig. 3.2 Original Image of Microstructure with corresponding Mask

3.3 Model Architecture

The study explored three distinct model architectures—custom CNN, Xception with transfer learning, and U-Net with transfer learning—to predict grain boundaries in microstructure images, each offering unique strategies for feature extraction and boundary segmentation.

1. **Custom CNN**: The first approach was a CNN built from scratch, tailored specifically to identify patterns in microstructure images. This CNN included multiple convolutional layers

to capture edge details and abstract features, followed by pooling layers to reduce spatial dimensions and fully connected layers for boundary prediction. While efficient in detecting simpler patterns, the custom CNN faced challenges with fine boundary details, requiring more sophisticated architectures for complex segmentation tasks.

2. **Xception Model with Transfer Learning**: The Xception model, a pre-trained deep network, was adapted using transfer learning to enhance feature extraction. Its depthwise separable convolutions allowed for efficient learning of complex structures within the grain boundaries. Transfer learning enabled the model to leverage pre-learned features from large datasets, significantly boosting performance and precision in boundary delineation, especially for challenging microstructure variations.
3. **U-Net with Transfer Learning**: U-Net, an encoder-decoder network, was fine-tuned to excel in boundary segmentation tasks. Skip connections between encoder and decoder layers preserved spatial resolution, making U-Net particularly effective at capturing fine and complex boundary details. This architecture ultimately outperformed the other models, achieving the most accurate and detailed grain boundary predictions due to its robust feature extraction and spatial accuracy.

3.3.1 Convolutional Neural Network (CNN) from Scratch

The first model developed was a custom CNN built from scratch. This model was designed to learn low- and high-level features through a series of convolutional, activation, and pooling layers. The main components of this architecture are outlined below:

1. **Convolutional Layers**: The CNN includes multiple convolutional layers with small 3x3 filters, designed to capture edges, textures, and patterns relevant to boundary detection.
    - Each layer increases in depth (number of filters) to extract a diverse set of features from low-level edge information to higher-level structural details.
2. **Activation Functions**: ReLU (Rectified Linear Unit) activation functions were applied after each convolution to introduce non-linearity, helping the model capture complex boundary structures.
3. **Pooling Layers**: Max-pooling layers, with a 2x2 filter, were used for down-sampling. By reducing the spatial dimensions of feature maps, these layers enabled the model to focus on the most prominent boundary features while reducing computational load.

4. **Fully Connected Layers**: Two dense layers at the end of the model aggregated learned features, followed by a final output layer with sigmoid activation, designed to produce a binary output for boundary and non-boundary regions.

Despite capturing essential boundary features, this CNN's limited depth and lack of pre-trained knowledge meant it struggled with fine-grained boundary details, particularly in regions with overlapping textures. The entire implementation code can be found in Ref. [3].

3.3.2 Transfer Learning with Xception Model

The second approach involved using the Xception model, a deep CNN architecture pre-trained on the ImageNet dataset, leveraging its efficient depthwise separable convolutions for detailed feature extraction.

1. **Pre-trained Encoder**: The pre-trained Xception model, known for its depthwise separable convolutional layers, served as the encoder. These layers perform spatial and depth-wise convolutions separately, significantly reducing parameter count and enhancing feature extraction capacity.

2. **Feature Extraction**: The Xception model's deep architecture enabled it to capture complex textural patterns in the microstructure images. By leveraging features from early layers, which detect general textures, to deeper layers, which capture fine-grained details, the model was capable of identifying subtle boundary characteristics.

3. **Custom Decoder**: A decoder was appended to the encoder to transform high-level features back to the spatial resolution of the original image. The decoder consisted of:

    - **Up-sampling Layers**: Transpose convolutions were used to progressively up-sample feature maps.

    - **Convolutional Layers**: Additional convolutional layers refined the boundary predictions at each up-sampling stage.

    - **Output Layer**: A final sigmoid-activated layer produced a boundary probability map, classifying each pixel as boundary or non-boundary.

By using transfer learning, the Xception-based model demonstrated improved boundary localization, although it occasionally lacked spatial continuity along edges due to the absence of skip connections found in fully convolutional architectures like U-Net. The entire implementation code can be found in Ref. [5].

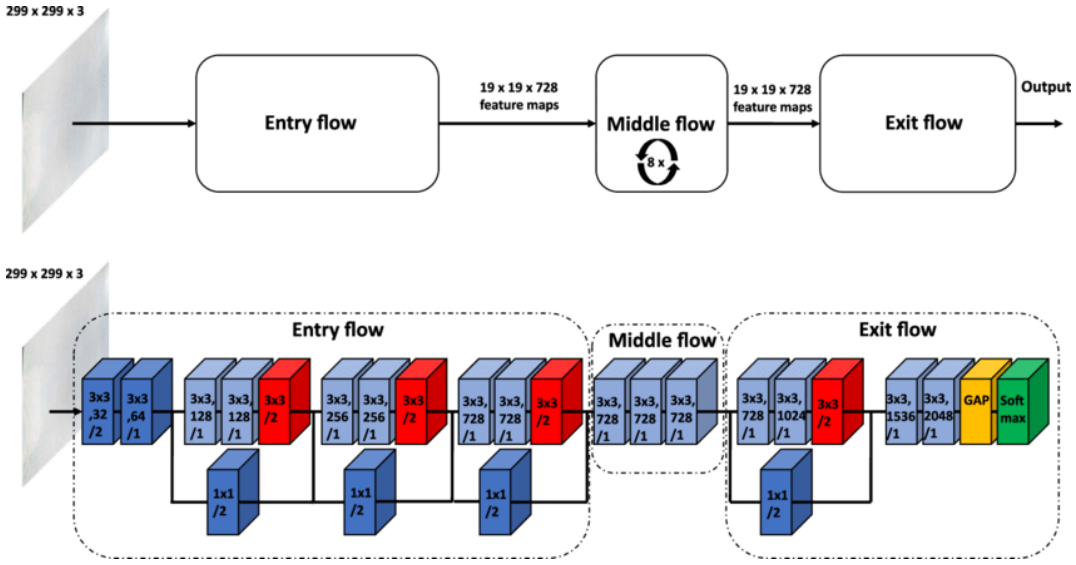


Fig.3.3 Xception Model Architecture

3.3.3 U-Net with Transfer Learning

The third and most successful approach was based on the U-Net architecture, a fully convolutional network designed specifically for segmentation tasks. U-Net's encoder-decoder structure and skip connections enabled it to capture and retain boundary details with high accuracy.

1. **Encoder**: A pre-trained model was used as the encoder in U-Net to capture high-level semantic information. Layers in the encoder progressively reduced spatial resolution through pooling operations, extracting features from coarse grain patterns to fine textures.

2. **Skip Connections**: A unique feature of U-Net is its skip connections, which pass feature maps directly from encoder layers to corresponding decoder layers. These connections preserve fine details, enhancing the model's capacity to detect boundaries with high spatial precision.

3. **Decoder**: The decoder up-samples the encoder's feature maps back to the original image resolution.

    - Each up-sampling step involves concatenating the corresponding encoder feature maps (enabled by skip connections) with the up-sampled feature maps from the previous decoder layer.

    - Convolutional layers after each concatenation refine the feature maps, focusing on both local boundary details and global structure.

4. **Output Layer**: The final layer is a sigmoid-activated layer, which provides pixel-wise boundary probabilities, outputting a mask that identifies boundary pixels.

Due to its skip connections and full-resolution outputs, U-Net achieved superior boundary delineation, especially for complex boundary geometries, making it the most effective approach for this task. The entire implementation code can be found in Ref. [4].

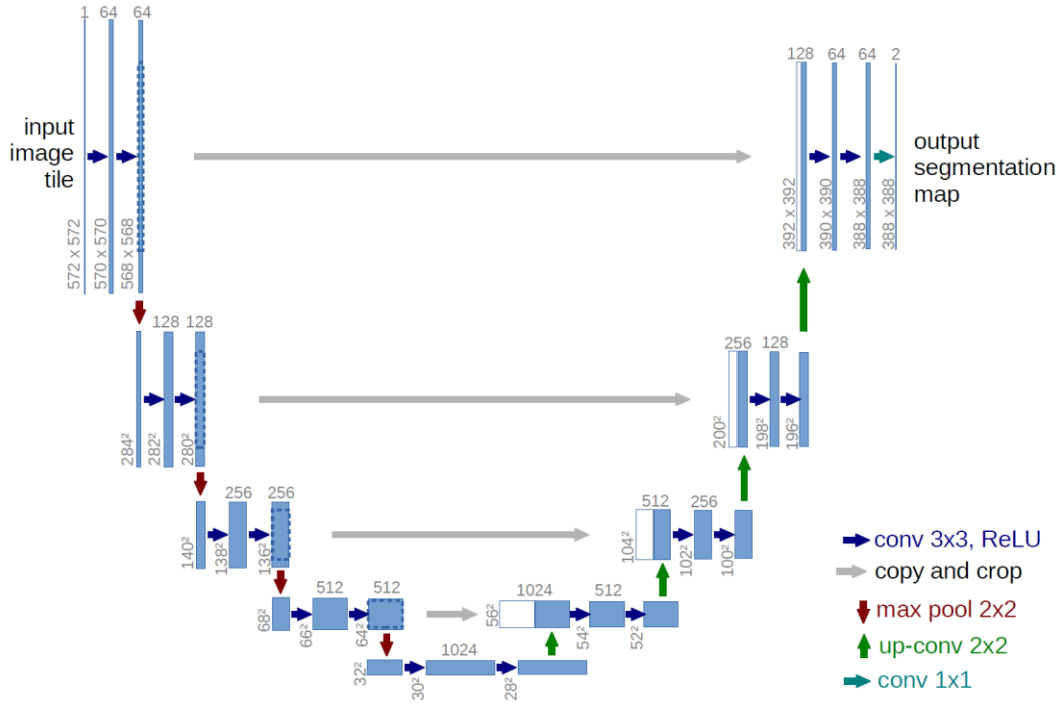


Fig. 3.4 U-Net Architecture

3.4 Training Process

Each model was trained on the dataset using a carefully designed training process:

1. **Loss Function**: Binary cross-entropy was initially used for all models. However, to enhance the U-Net model's performance in boundary delineation, a combination of binary cross-entropy and Dice loss was experimented with, as Dice loss optimizes the overlap between predicted and true boundaries.

2. **Optimizer**: The Adam optimizer was chosen for its adaptive learning rates, which help achieve faster convergence and stabilize training.

3. **Data Augmentation**: Various augmentations were applied to prevent overfitting and increase model robustness:

    - **Rotation and Flipping**: Random rotations and flips preserved orientation diversity.

    - **Random Cropping and Scaling**: These adjustments simulated boundary variations, enhancing the model's ability to generalize.

4. **Training Configuration**: Each model was trained for a set number of epochs, with early stopping based on validation loss to avoid overfitting.

3.5 Preparation of Custom Dataset for future works

The dataset preparation process was carefully designed to obtain high-quality images of grain boundaries in 316L stainless steel samples. Each stage of the process, from sample preparation to final image acquisition, required careful handling and adherence to protocols to ensure that the images captured were suitable for use in our deep learning model.

1. Sample Material: 316L Stainless Steel

We selected 316L stainless steel as our sample material for this study. This stainless steel alloy is widely used in industries that require resistance to corrosion and extreme temperatures, making it ideal for observing grain structures. The 316L alloy contains molybdenum, which enhances corrosion resistance, as well as chromium, nickel, and low carbon content. The low carbon content in 316L stainless steel helps minimize carbide precipitation, which is beneficial for the etching process.

2. Sample Cutting via EDM (Electrical Discharge Machining)

To prepare samples of appropriate size and shape, we used Electrical Discharge Machining (EDM). EDM is a non-contact process that employs electrical discharges to remove material from a workpiece, making it ideal for cutting complex and precise shapes without applying mechanical force. The EDM process allowed us to cut samples precisely without introducing any mechanical stress or deformation to the grain structure of the material, which is crucial for obtaining undistorted grain boundary images.

3. Mold Preparation Using Hot Molding

Following cutting, the sample was embedded in a phenolic resin block using hot molding. This process involved placing the sample in a mounting press along with phenolic powder, which, under heat and pressure, formed a solid mount around the sample. The phenolic resin protected the sample's edges during grinding and polishing and provided ease of handling. Hot molding is particularly advantageous for brittle or small samples, as it stabilizes the sample for further processing.

4. Grinding and Polishing

Once the sample was mounted, we moved on to the grinding and polishing stage, a critical process for revealing the grain boundaries in the metal. During grinding, abrasive particles were used to remove surface irregularities and flatten the sample surface. We began with coarse abrasive discs, gradually moving to finer ones to create a smooth, scratch-free surface. This multi-step process involved different grades of silicon carbide paper, typically moving from coarser (e.g., 240 grit) to finer grits (e.g., 2500 grit).

Following grinding, the sample was polished using alumina suspensions to achieve a mirror-like finish. The final polishing stage used a very fine polishing cloth with alumina suspensions of less than 3-micron size to ensure the sample surface was completely smooth. Polishing is essential as it minimizes scratches and defects on the surface, which would otherwise interfere with the etching and imaging stages.

5. Etching Process

With a polished sample surface ready, we proceeded to the etching step. Etching is a controlled corrosion process that selectively dissolves material at grain boundaries, enhancing their visibility. The etching solution used was a custom mix designed for stainless steel, containing:

- Hydrochloric Acid (HCl): 10 ml
- Nitric Acid (HNO3): 5 ml
- Glycerol: 10 ml

The inclusion of glycerol in the etchant helps in slowing down the etching reaction, providing better control and yielding a more refined view of the grain structure. This etching solution effectively reveals the grain boundaries by selectively attacking the material at the boundaries, making them appear as dark lines under a microscope. The sample was immersed in this etchant for precisely 180 seconds. Strict timing was essential in this step to avoid over-etching, which could obscure finer grain boundary details.

Fig. 3.5 Preparation of Mold (316L stainless steel)

6. Image Capturing and Final Dataset Preparation

After etching, the sample was promptly rinsed and dried to halt the reaction. High-resolution images of the grain boundaries were then captured at 100x magnification using a metallurgical microscope under controlled lighting conditions. Each image was carefully reviewed to ensure it met quality standards for our deep learning application. The final dataset, consisting of these high-quality, 100x-magnified images, displayed grain boundaries with well-defined contrast and structure, making it highly suitable for training and evaluating boundary detection models.

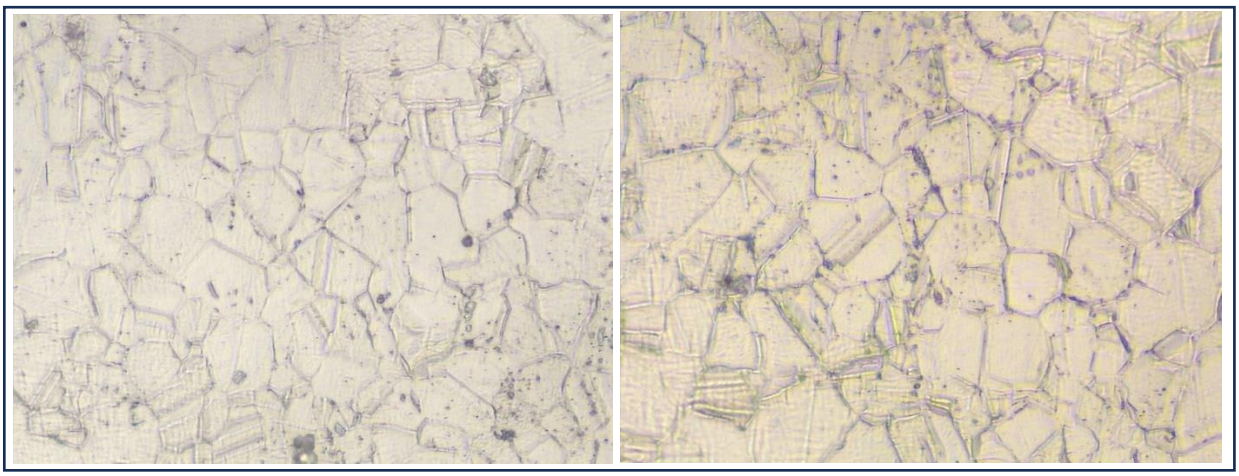


Fig. 3.6 Image Capturing for Dataset

3.5 Summary

In this chapter, we described the three deep learning-based architectures used to predict grain boundaries in microstructure images. The custom CNN offered a foundational approach, though limited by its simplicity. The Xception model, with transfer learning, leveraged pre-trained knowledge for enhanced feature extraction, leading to improved boundary predictions. Finally, the U-Net with transfer learning demonstrated the highest performance, owing to its encoder-decoder structure and skip connections, which preserved spatial details critical for boundary delineation. This analysis informed our selection of U-Net as the optimal model for precise boundary detection in microstructure images.

CHAPTER 4

RESULTS AND DISCUSSIONS

In this section, we will discuss the results obtained from various methods employed to separate grain boundaries from grain images and generate the corresponding masks. Several image processing techniques, machine learning models, and architectures were tested during the course of this study to optimize the boundary detection process. However, many of the methods tested did not provide satisfactory results. These methods were chosen based on existing literature and included OpenCV-based approaches such as edge detection, clustering techniques, and contour plotting. Despite various efforts, the accuracy of the predictions was not up to the expected standards. As a result, we shifted to a more manual and deep learning-based approach for mask creation.

4.1 Utilization of various Computer Vision & ML tools:

4.1.1 Edge Detection

Edge detection is a fundamental method for identifying the boundaries of objects within an image. In our case, we used the Canny edge detection algorithm to identify grain boundaries. The Canny edge detector is a multi-step process involving noise reduction, gradient calculation, non-maximum suppression, and edge tracing. Despite its ability to detect edges, the Canny method produced poor results when applied to grain images. The boundaries it identified were often too rough and fragmented, leading to incomplete or noisy masks.

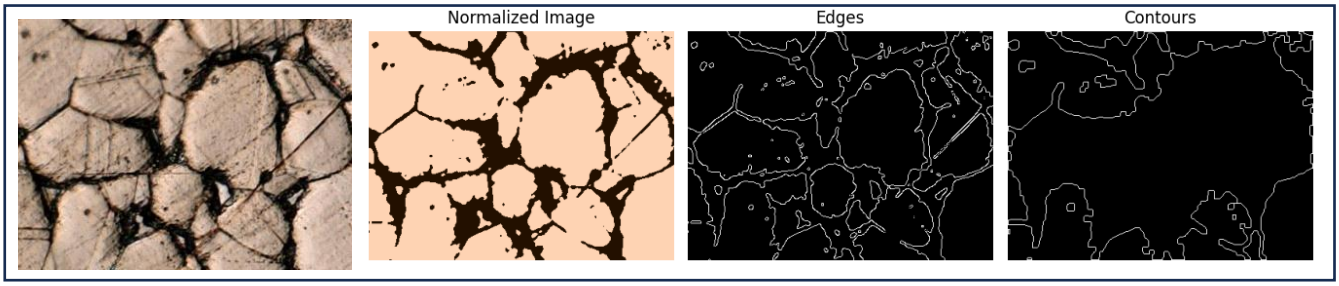
This limitation is partly due to the fact that Canny edge detection depends on a set of thresholds, which need to be tuned for each specific image type. In our case, the variability in grain structure and image quality made it difficult to find an optimal set of thresholds. The boundaries detected were often too sparse or over-saturated, missing out on smaller, finer grain structures that were critical for accurate segmentation.

The final output masks were not as accurate as expected, and the edges identified by the algorithm did not match the true boundaries of the grains.

4.1.2 K-Means Clustering and OpenCV Contour Extraction

The next approach we tried was to use K-means clustering to group pixels into two clusters: one representing the grain boundaries (dark regions) and the other for the non-boundary regions. The dark regions, typically having pixel values between 240-255, were assigned to one cluster, while other regions were grouped into the second cluster. Afterward, we extracted the grain boundaries by isolating the first cluster. However, this method did not produce satisfactory results, as the contours of the resulting mask were not accurate, and many grain boundaries were either missed or falsely detected.

To address this, we attempted contour extraction using OpenCV, which involves detecting boundaries by identifying the transitions between regions of different intensity levels. While this technique is generally effective for boundary extraction in simple images, it failed in the context of grain images. The results were disappointing due to the high noise levels in the grain structure, leading to irregular and fragmented contours in the extracted masks. Additionally, when we attempted to extrapolate the contours to refine the boundaries, the result was overly smoothed or left with gaps, making the predicted mask inconsistent with the actual grain boundaries.



(a) (b) (c) (d)

Fig. 4.1 (a) Original image (b) Clustering results (c) Canny edge detection (d) OpenCV contour extrapolation

4.1.3 Manual Annotation for Supervised Learning

Given the limitations of the OpenCV-based approaches, we decided to switch to a more reliable method for creating masks, which involved manually annotating the grain boundaries. Inspired by a research paper titled "*Grain and Grain Boundary Segmentation Using Machine Learning with Real and Generated Datasets*", which employed hand annotation for training deep learning models, we manually created the masks using MS Paint. This process involved labour-intensive work to annotate the grain boundaries pixel by pixel, ensuring that the mask accurately represented the true boundaries in the images.

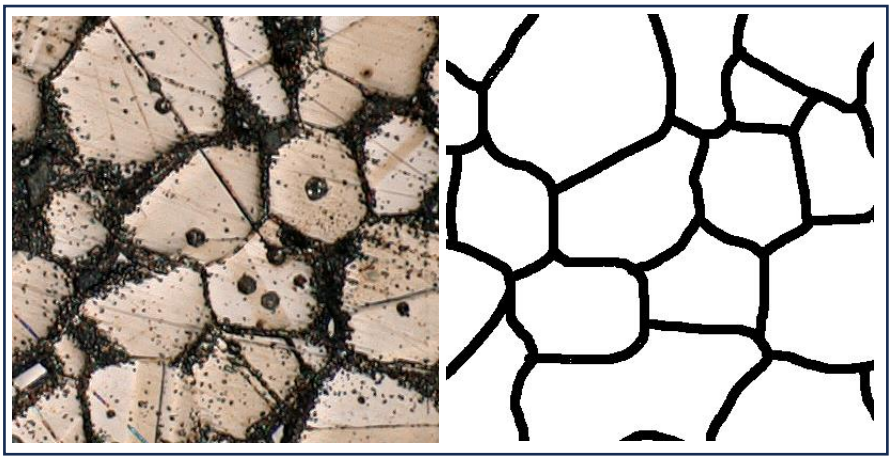


Fig. 4.2 Microstructure image and its manual annotated mask

This manual labelling provided us with high-quality masks that could serve as the ground truth for supervised learning. We used these masks to train our deep learning models, with the aim of improving the accuracy of boundary predictions. The following models were tested:

4.1.4 Model 1: U-Net Architecture without Transfer Learning

The first deep learning model we implemented was a U-Net-based architecture. This self-made convolutional neural network (CNN) did not use transfer learning and was trained from scratch on the grain images and their manually annotated masks. The U-Net model, known for its encoder-decoder structure with skip connections, was chosen due to its effectiveness in image segmentation tasks.

After training the model for a few epochs, we plotted the accuracy vs. epochs, loss vs. epochs, and Dice coefficient vs. epochs. The accuracy and Dice coefficient plots indicated that the model was struggling to improve during training, and the loss did not decrease significantly. The predicted mask comparison with the actual mask showed that the model was unable to capture the grain boundaries accurately. The predictions were blurry, with many areas missing or incorrectly identified as part of the boundary. These results were disappointing, suggesting that the model was not performing well without transfer learning.

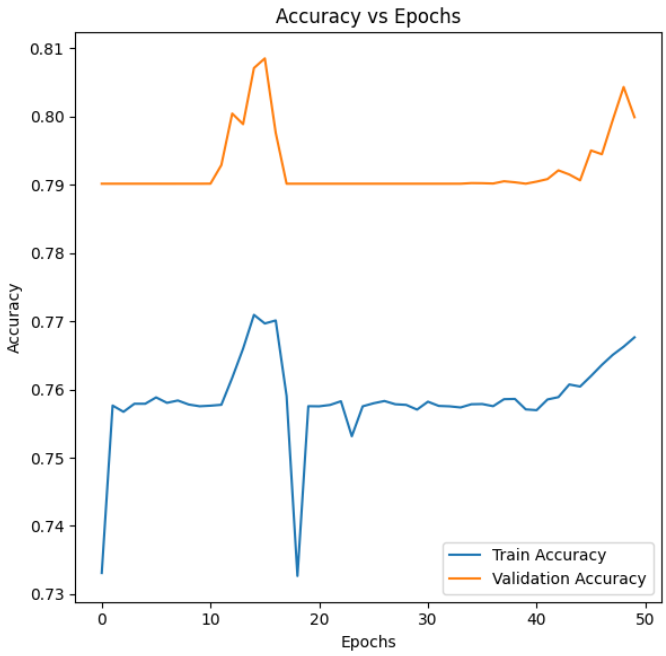


Fig. 4.3 Accuracy vs Epochs Plot using U-Net

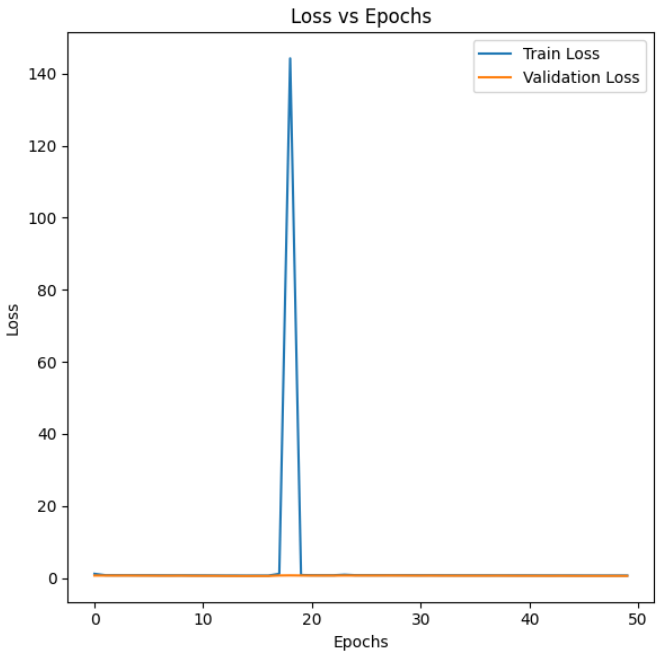


Fig.4.4 Loss vs Epochs Plot using U-Net

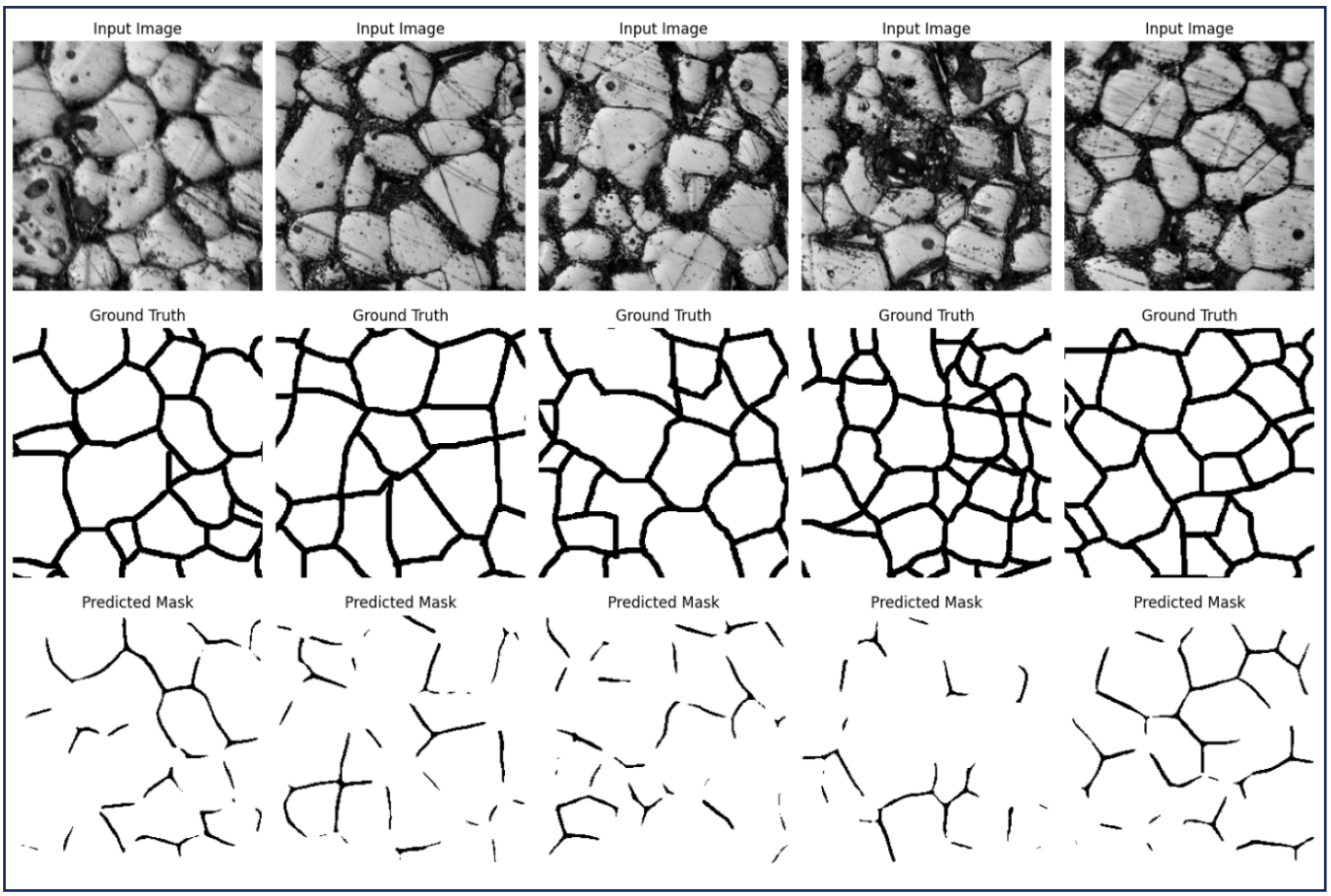


Fig.4.5 Input image with the Ground Truth and the corresponding Predicted Mask using U-Net

4.1.5 Model 2: Xception Architecture with Transfer Learning

In the second experiment, we employed the Xception architecture, utilizing transfer learning. The Xception model, which is a deep convolutional network based on depth wise separable convolutions, was pre-trained on a large dataset (ImageNet) and fine-tuned on our grain boundary dataset. Transfer learning has been proven to be effective in improving performance, especially when working with smaller datasets, by leveraging pre-learned features from a larger corpus of data.

Once again, we plotted the accuracy vs. epochs, loss vs. epochs, and Dice coefficient vs. epochs. The results were similar to the first model—while the training loss decreased, the accuracy did not improve as expected. The predicted masks did show some improvement compared to the U-Net model, but they still lacked sharpness and precision. Upon further inspection, it appeared that the model may have been overfitting to the training data, especially given the complexity of the Xception architecture. This overfitting likely resulted from a limited dataset size and the model's high capacity, leading to poor generalization.

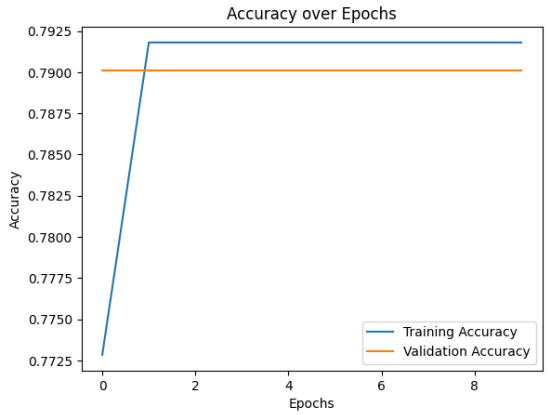


Fig.4.6 Accuracy vs Epochs using Xception

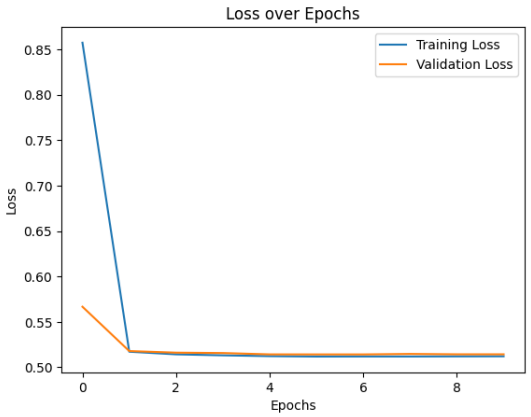


Fig.4.7 Loss vs Epochs using Xception

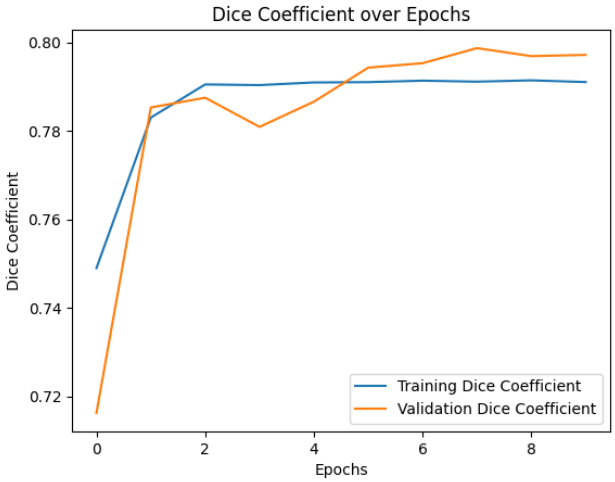
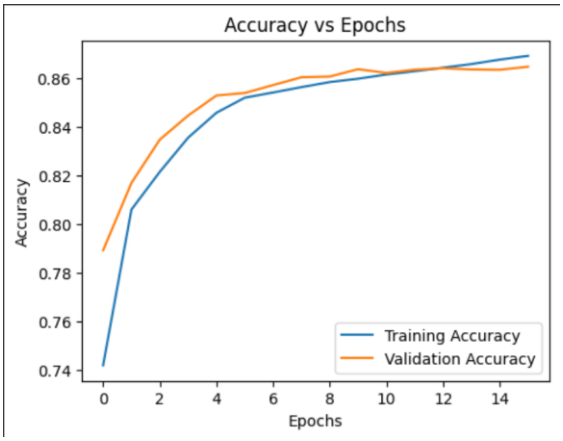


Fig.4.8 Dice Coefficient vs Epochs using Xception

4.1.6 Model 3: U-Net Architecture with Transfer Learning

For the final model, we employed a U-Net architecture with transfer learning. We fine-tuned the U-Net model, leveraging the pre-trained weights from ImageNet, to train on the manually annotated grain images and masks. The U-Net model's encoder-decoder structure, when combined with transfer learning, allowed the model to learn both low-level and high-level features more effectively.

The results were significantly better compared to the previous models. The accuracy reached an impressive 87%, and the Dice coefficient also improved, showing a better overlap between the predicted and actual masks. The predicted masks were more accurate and aligned well with the grain boundaries, demonstrating that this architecture was the most effective in predicting grain boundaries. The model successfully captured fine details and eliminated much of the noise, leading to high-quality boundary predictions.



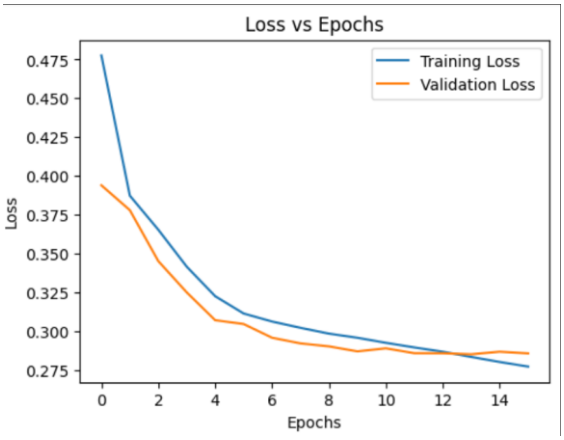


Fig.4.9 Accuracy vs Epochs for Transfer Learning(U-Net) **Fig.4.10** Loss vs Epochs for Transfer Learning (U-net)

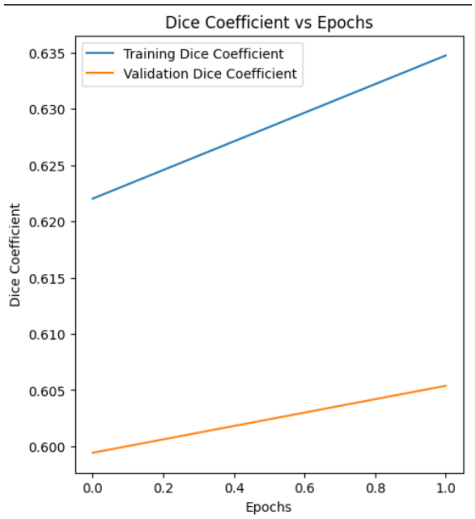


Fig.4.11 Dice Coefficient vs Epochs for Transfer Learning (U-net)

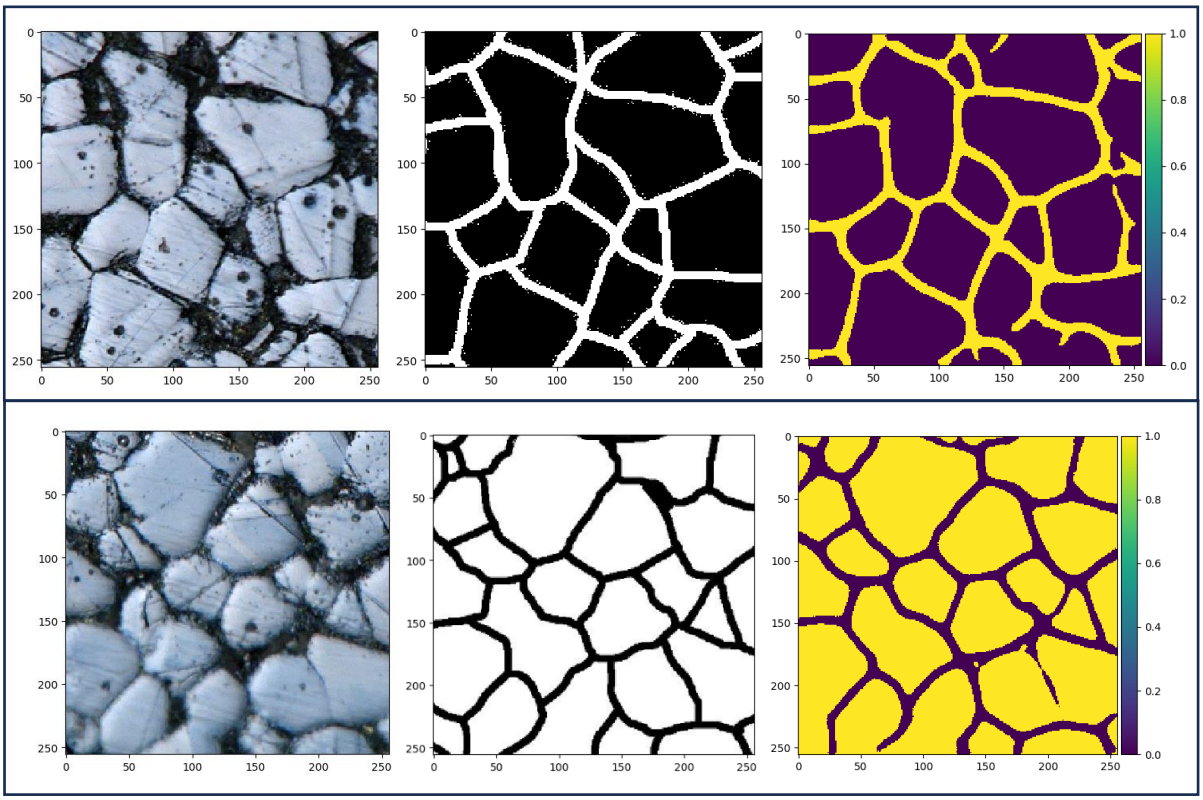


Fig.4.12 Input image with the Label mask and the corresponding Predicted Mask using U-Net

4.2 Conclusion

The results of this study demonstrate the limitations of traditional image processing methods such as edge detection and clustering when applied to grain boundary detection. While these methods are effective in some contexts, they failed to deliver satisfactory results in our specific case due to the complexity and variability of the grain images. Manual annotation and deep learning models, particularly U-Net with transfer learning, proved to be far more effective in generating accurate masks for supervised learning.

The U-Net model with transfer learning produced the best results, achieving an accuracy of 87% and generating masks that closely matched the actual grain boundaries. This suggests that deep learning models, especially those that utilize transfer learning, are a promising approach for grain boundary detection and segmentation tasks. Future work could involve further fine-tuning of these models, exploring more complex architectures, and increasing the dataset size to enhance performance even further.

In conclusion, while early attempts using traditional image processing techniques were unsuccessful, the adoption of deep learning-based approaches, particularly with manual annotations for training, resulted in substantial improvements in the prediction accuracy. This highlights the potential of deep learning for more accurate and efficient grain boundary detection in microstructure images.

CHAPTER5

CONCLUSIONS & FUTURE SCOPES

5.1 Key Findings

5.1.1 Challenges with Traditional Image Processing Methods

Traditional image processing techniques, including edge detection, clustering, and contour plotting, proved to be inadequate for grain boundary detection in our study. While these methods, such as Canny edge detection and k-means clustering, are effective for simpler boundary detection tasks, they struggled with the grain images due to the complex textures, varied grain shapes, and noise. Our findings confirmed that these methods failed to generate well-defined boundaries, indicating their limitations in handling the intricate patterns of microstructure images.

5.1.2 Limitations of Simple Mask Generation Techniques

Attempting to generate masks by clustering pixels based on brightness levels also failed to produce satisfactory results. Specifically, separating darker pixels for boundary detection resulted in broken and incomplete contours. The boundary details were either missing or distorted, which led to inaccuracies in the generated masks. Even techniques involving OpenCV contour extraction and extrapolation did not yield consistent results, further emphasizing the need for a more robust approach for our dataset.

5.1.3 Effectiveness of Manual Annotation

Given the limitations of automatic boundary detection methods, manual annotation was employed. Inspired by a research study, "Grain and Grain Boundary Segmentation Using Machine Learning with Real and Generated Datasets," this manual segmentation approach

involved hand-tracing the grain boundaries to create high-quality masks. This labor-intensive process significantly improved the dataset's accuracy, serving as a reliable basis for training deep learning models in the next steps. Manual annotation provided a strong foundation for generating precise boundary labels that would enhance model learning.

5.1.4 Deep Learning Performance and U-Net with Transfer Learning

Among the deep learning approaches explored, the U-Net model with transfer learning outperformed all others. It achieved a promising 87% accuracy, producing masks that were highly aligned with the actual grain boundaries. This indicates that U-Net's architecture, which is designed for semantic segmentation tasks, combined with transfer learning, was well-suited to the complex patterns of grain boundaries. The model's ability to generalize effectively to various grain boundary patterns demonstrated that deep learning architectures with sufficient training can significantly enhance segmentation accuracy.

5.1.5 Potential for Future Model Enhancements

The success of U-Net with transfer learning highlights the potential for further improvements. For future work, expanding the dataset, fine-tuning hyperparameters, and experimenting with alternative loss functions—such as those that take coordinates instead of pixel-based errors—could yield even better performance. Additionally, using more advanced architectures or hybrid models designed for segmentation tasks could further enhance boundary detection capabilities.

5.1.6 Importance of High-Quality Labels for Training

This study emphasizes the importance of accurate labels, as manually annotated masks substantially improved model accuracy. Quality annotations directly impacted model training outcomes, showing that well-prepared datasets are crucial in achieving high segmentation performance. This finding underscores the value of investing time in label quality to maximize the predictive accuracy of deep learning models in boundary detection tasks.

In summary, our key findings highlight the shift from traditional methods to deep learning solutions as a vital step in achieving reliable grain boundary detection. The transition underscores the importance of manual annotation, the benefits of U-Net with transfer learning, and the potential for future model refinements, paving the way for more accurate and efficient grain boundary segmentation in microstructure images.

5.2 Future Scopes

Building on this research, there are several planned advancements that aim to optimize the accuracy, effectiveness, and adaptability of grain boundary detection and segmentation within our custom-developed dataset. These future directions include:

5.2.1 Application of Manual Annotation and Segmentation on Custom Dataset:

With our dataset in place, the next step involves detailed hand annotation, where each grain boundary will be manually segmented to create highly accurate labels. This manual segmentation process will enable precise mask generation for training purposes, ensuring that our model learns from well-defined boundaries specific to our dataset's grain structure. By tailoring these masks to our unique dataset, we anticipate enhanced model performance in recognizing and predicting grain boundaries.

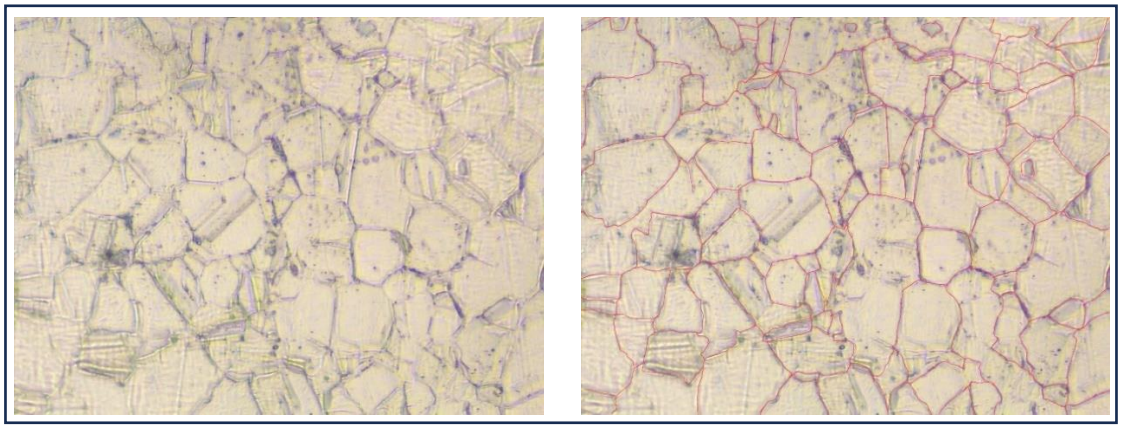


Fig. 5.1 Microstructure with corresponding Hand Annotation

5.2.2 Quantitative Parameter Extraction:

Once the model achieves satisfactory boundary predictions, various metallurgical parameters will be calculated to evaluate both the material and the model's performance. These include:

- **Accuracy Assessment**: Utilizing established methods such as the line intercept method to validate boundary prediction accuracy against ground truth data.
- **Average Grain Size Calculation**: Calculating the mean grain size from segmented regions to correlate with physical properties.
- **Area Analysis**: Quantifying the segmented regions to assess grain area distribution, providing insights into material characteristics and model effectiveness in boundary delineation.

5.2.3 Alternative Training Methods for Enhanced Boundary Precision:

Beyond pixel-based segmentation, future implementations may include training methods that leverage contour-based approaches. Extracting and optimizing the contour coordinates directly, rather than relying solely on pixel-based optimization, could offer finer boundary details, especially useful for complex grain structures. Exploring error functions that utilize coordinate-based evaluations rather than pixel-wise accuracy could further refine the model's capability to interpret continuous boundary lines more precisely.

5.2.4 Exploration of Advanced Model Architectures and Evaluation Metrics:

Beyond our current architectures, experimenting with new model designs tailored to boundary prediction could yield significant improvements. For example:

- **Customized Loss Functions**: Developing or implementing loss functions that penalize boundary misalignments based on contour accuracy rather than just pixel similarity. This could allow for a higher tolerance to minor inaccuracies within grains while focusing on perfecting boundary prediction.
- **Experimenting with Architectures Optimized for Contour-Based Learning**: Certain architectures might be better suited to learning structural boundaries, such as networks designed for geometric and coordinate-based tasks. Exploring these models could result in more reliable and detailed grain boundary predictions.

5.2.5 Broadening Model Applicability Through Transferable Training:

While the initial focus is on 316L stainless steel, the developed techniques and models could later be generalized to work on different materials, broadening the model's applicability within material science. Expanding to new materials could confirm the robustness of the approach and highlight its potential as a versatile tool for analyzing varied grain structures.

This focused future work will bring more robust, detailed, and adaptable grain boundary detection models, capable of producing high-precision results and providing valuable quantitative metrics for materials science research and industrial applications.

References

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, University of Toronto, ImageNet Classification with Deep Convolutional Neural Networks

[2] Arun Baskarana, Genevieve Kanea, Krista Biggsb, Robert Hulla, Daniel Lewisa, Adaptive characterization of microstructure dataset using a two stage machine learning approach

[3] https://github.com/nidhisingh2309/BTP-Sem-7/blob/main/Scratch_CNN.ipynb

[4] https://github.com/nidhisingh2309/BTP-Sem-7/blob/main/UNet.ipynb

[5] https://github.com/nidhisingh2309/BTP-Sem-7/blob/main/XceptionRe.ipynb

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition.

[7] Lun Che, Zhongping He, Kaiyuan Zheng, Tianyu Si, Meiling Ge, Hong Cheng, Lingrong Zeng, Deep learning in alloy material microstructures: Application and prospects, College of Mechanical Engineering, Chengdu university, China

[8] Peter Warrena, Nandhini Rajub, Abhilash Prasadb, Md Shahjahan Hossainb, Ramesh Subramanianc, Jayanta Kapatb, Navin Manjoorana, Ranajay Ghoshb, Grain and grain boundary segmentation using machine learning with real and generated datasets.

[9] P. Warren, ExONE stainless steel 316L grains 500X, 2023, https://www.kaggle.com/datasets/peterwarren/exone-stainless-steel-316l-grains-500x.

[10] Seyed Majid Azimi, Dominik Britz, Michael Engstler, Mario Fritz1 & Frank Mücklich, Advanced Steel Microstructural Classification by Deep Learning Methods, Received: 29 June 2017, Accepted: 10 January 2018, Published online : 01 February 2018

[11] Yann Lecun, Member, IEEE, L´ Eon Bottou, Yoshua Bengio, and Patrick Haffner, Gradient-Based Learning Applied to Document Recognition

[12] Yongfeng Li , Shuhui Li, Deep learning based phase transformation model for the prediction of microstructure and mechanical properties of hot-stamped parts, state Key Laboratory of Mechanical System and Vibration, Shanghai Key Laboratory of Digital Manufacture for Thin-walled Structures, Shanghai Jiao Tong University, Shanghai, 200240, China Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning