MSS
MATERIALS SCIENCE AND SIMULATION

RUHR UNIVERSITÄT BOCHUM
RUB

σ-phase Particles Segmentation in CrMnFeCoNi High Entropy Alloys using Machine Learning

Project Work

by

Zeki Ayrildi

A report submitted in partial fulfillment of the requirements of Materials Science and Simulation for obtaining the academic degree Master of Science (M.Sc.)

completed at
Materials Informatics and Data Science - Interdisciplinary Centre for Advanced Materials Simulation Ruhr Universität Bochum

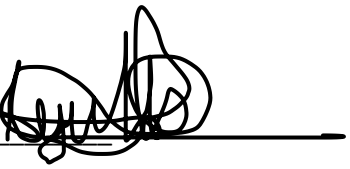
ICAMS
INTERDISCIPLINARY CENTRE FOR ADVANCED MATERIALS SIMULATION

Supervisor
Prof. Dr. Markus Stricker

Bochum, November 4, 2023

Statutory Declaration

I declare that I completed this work on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this, nor a similar work, has been published or presented to an examination committee.

04.11.2023, 

Date, Signature

Contents

Abstract

The interior structure of a substance is referred to as its microstructure. It records a material's origin and specifies all of its mechanical and chemical properties, such as the σ phase, which is an intermetallic compound generated in high entropy alloys that affects material properties. The microstructural segmentation of the σ phase is difficult because the fcc matrix and σ phase have overlapping grayscale distributions, making image analysis tools difficult to analyze and hand classification time-consuming. Machine Learning approaches have recently demonstrated significant performance in Computer Vision applications by learning features from data alongside the classification process. In our project, we employed a sequential machine-learning approach. Initially, we extracted features from backscattered electron images for training our machine-learning model, which was then used for segmentation. Subsequently, we utilized deep learning techniques, specifically the U-Net architecture and transfer learning, for the semantic segmentation of the σ phase. In this phase, features were automatically extracted from the backscattered electron images, and the model performed the segmentation. Ultimately, we achieved successful segmentation of σ phases, showing a strong correlation in terms of shape and size.

1 Introduction

The mechanical characteristics of a material are largely influenced by its microstructure. Essentially, how the material behaves is closely tied to the arrangement, form, and dimensions of the phases present within its microstructure. As a result, accurately identifying these microstructures is critical. The appearance of a material's microstructure can vary widely, affected by numerous factors. These include the elements mixed into the alloy, the rolling process, how quickly it cools, the heat treatments, and other subsequent treatments. The specific production processes and conditions determine the unique components that make up the microstructure [1].

Vander Voort utilized Light Optical Microscopy (LOM), a method that remains the gold standard in material science for classifying microstructures in many academic and industrial settings [2]. His approach primarily outlined procedures that allowed experts to determine the microstructure's class. Earlier, Velichko and colleagues introduced a technique that employed data mining. They extracted morphological characteristics and then classified these features in cast iron using Support Vector Machines (SVMs), a renowned machine learning method [3]. In a more recent study, Pauly and his team adopted a similar strategy. They used a dataset of steel, obtained through both scanning electron microscope (SEM) and LOM imaging techniques, which was also employed in this research. However, their method achieved only 48.89 % accuracy in classifying microstructures across four categories. This limited success was attributed to the intricate substructures and the features not being distinct enough [4].

Deep learning, a branch of advanced computing, has been employed for tasks such as identifying objects in images and segmenting images into meaningful sections. One notable deep learning model is AlexNet, developed by Alex Krizhevsky and colleagues. This model gained significant attention after winning the ImageNet Challenge in 2012 [5], a major competition in the tech community. What made AlexNet special was its significant improvement in accuracy. Following this, another model named VGGNet, which had more layers than AlexNet, achieved even better performance [6]. Later, Fully Convolutional Neural Networks (FCNNs) introduced by Long and his team adapted models like AlexNet for semantic segmentation. Currently, FCNNs and their derivatives are among the top tools for semantic segmentation [7].

Semantic segmentation is a computer vision task in which the objective is to classify each pixel of an image into a specific category or class. Unlike simple image classification, where the entire image is assigned one label, semantic segmentation provides detailed segmentation by categorizing each pixel. This results in a detailed, pixel wise mask for each object in the image[8].CrMnFeCoNi is a high entropy alloy (HEA) which is a class of materials that are composed of multiple principal elements in near-equal atomic percentages. These alloys have attracted significant attention due to their unique combination of properties, which can include high strength, ductility, and corrosion resistance, among others. The σ phase is an intermetallic compound that can form in many multi component alloy systems, including high entropy alloys. The formation of the σ phase can have significant implications for the properties of the alloy. Laplanche et al. described the methodology used to quantify the surface area fraction of the σ phase in a sample using ImageJ software, based on back scattered electron (BSE) micrographs. The two phases (σ and FCC matrix) have overlapping grayscale distributions. This means that they can appear quite similar in grayscale, making semantic segmentation difficult for automated image analysis software to distinguish

between them [9]. This motivation leads us to use Machine Learning methods which have recently grabbed the attention of scientists due to their strong ability to learn high-level features from raw input data. In this project work, two distinct methods were applied to analyze BSE micrographs. Initially, a traditional machine learning technique was employed, where specific characteristics were extracted from the images. Using a classifier, each individual pixel was then categorized based on these characteristics. Subsequently, a deep learning approach was introduced, utilizing a Convolutional Neural Network known as U-net to segment σ phase particles from BSE micrographs.

2 Methods

Getting precise characterizations of microstructures is key to understanding the relationship between structure, processing, and properties. However many tasks in microstructural analysis rely on human decision-making, introducing potential bias and making automation challenging. Since artificial intelligence techniques, like computer vision (CV) and machine learning (ML), can mimic human visual assessments, they offer a promising solution for carrying out these tasks objectively, independently, and efficiently. In this project, two distinct approaches were explored. The first is classical machine learning, where features are extracted manually using various filters, followed by pixel classification using the Random Forest Classifier. The second approach utilizes deep learning, where both feature extraction and pixel classification are handled by the neural network itself. Here, the objective is to distinguish and isolate σ phase particles from other phases present in back scattered electron scanning electron microscope images using Supervised Machine learning. Supervised learning in the context of binary semantic segmentation refers to the process where a machine learning model is trained to classify each pixel of an image into one of two classes, typically based on labeled training data. In binary semantic segmentation for this context, the aim is to split the micrograph into two clear segments: the σ phases represented by white pixels (foreground) and everything else that isn't the σ phase, which forms the background black pixels. The "supervised" aspect comes from the fact that the model is provided with the "ground truth" during training, which guides the learning process. The model's predictions are compared to the actual labels, and the model adjusts its internal parameters to improve its predictions over time. The ground truths, often referred to as masks, are obtained by manually segmenting images using the Matlab image segmenter app. It takes roughly 2.5 hours to manually segment each image. Throughout this project, Python was the programming language used.

The $Cr_{26}Mn_{20}Fe_{20}Co_{20}Ni_{14}$ high entropy alloy underwent various processes including vacuum induction melting, homogenization, rotary swaging, and recrystallization. Specimens were prepared via cutting, grounding, and polishing. Electropolishing was performed to prepare the specimens for microstructural examination. Samples were annealed at temperatures between 700 °C and 1000 °C for annealing times ranging from 1 hour to 1000 hours. Post-annealing, the specimens were cooled to room temperature. The microstructures of the prepared specimens were then examined using a scanning electron microscope of type Quanta FEI 650 ESEM. Backscattered electron micrographs were obtained at an acceleration voltage of 20 kV, a spot size of 6 µm, a working distance of 10 mm, and a scanning time of 30 µs.

2.1 Manually Feature Extraction and Random Forest Semantic Segmentation

To begin with, we adopted a structured approach using traditional machine learning for image processing. During the initial phase of our image analysis, we methodically applied a range of filtering techniques. We used these filters to get important features from the images. These features then helped train the machine learning model to segment the images correctly. The filters used are Gabor, Canny Edge, Robert, Sobel, Scharr, Prewitt, Gaussian, Median[10, 11, 12].

In this work, it is essential to classify each part of an image correctly, but doing this for every tiny part (pixel) of an image is tough, especially when we don't have a super powerful computer. So, a method is used by Shotton and his team, which they explained in their papers [13]. They used "random forests" to do this job efficiently. The Random Forest algorithm is a type of ensemble learning technique that is built upon multiple decision trees, as depicted in Figure 1. Compared to many neural networks, it is computationally more efficient[14]. So, a Random Forest operates as a collective of decision trees, each trained on a subset of the data and focusing on different aspects or features. When making a prediction, all the trees in the forest "vote" or give their input, and the majority decision is taken as the final output. This ensemble method leverages the strengths of multiple trees, making it more accurate and less prone to overfitting than a single tree. It is like consulting a group of experts instead of relying on just one. By following their steps, we tried to make pixel wise classifications for images.In a Random Forest classifier for binary image segmentation, each decision tree's task is to learn how to classify individual image pixels as either part of the foreground (object) or the background based on extracted image features. The decision trees independently make these binary predictions, and the final segmentation is determined by combining their outputs, typically through a majority voting scheme.

We began by setting up our environment by importing Numpy, OpenCV, pandas, sci-kit-learn, and scikit-image libraries and loading our target image, ensuring it was in grayscale. Features were extracted from the image using a combination of different filters and techniques.These features could include pixel values, color information, texture descriptors, gradient information, or any other characteristics that are informative for distinguishing between the σ phase and the background. For supervised learning, we incorporated manually labeled mask data corresponding to our original image. The data was then prepared and divided for training and testing. We chose a machine learning model, trained it, and evaluated its performance using a common accuracy metric. To ensure the model's practicality, it was saved and stored for future applications. In subsequent stages, we utilized the saved model to predict segment σ phase on the original image [15].

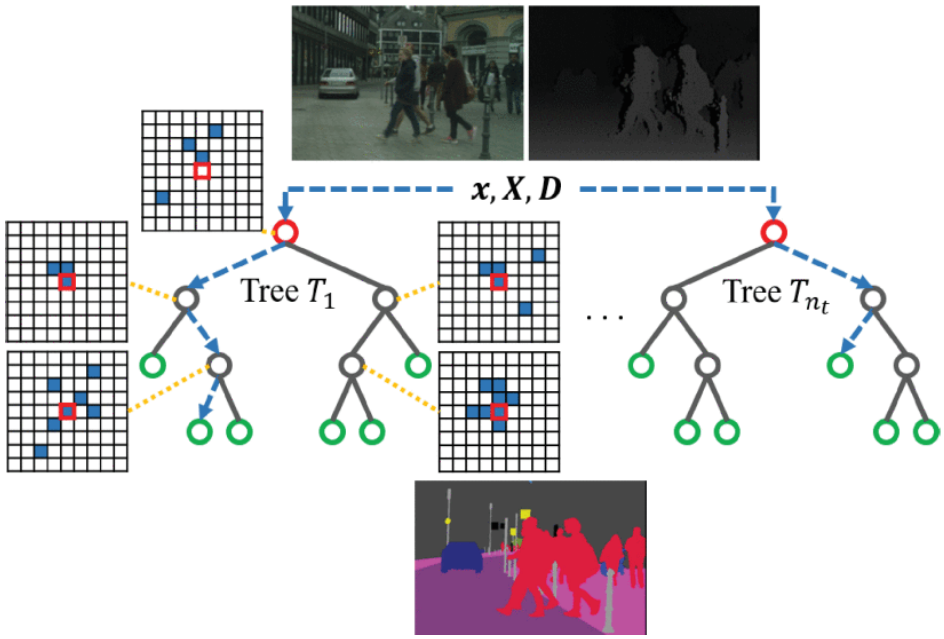


Figure 1: This depicts how the random forest uses learned patterns for semantic segmentation. The top images are the input images, which include a color picture and a depth map. The image at the bottom displays the desired output, which assigns a class label to each pixel. The grid-filled rectangles represent different learned patterns with varying shapes, densities, and importance. The red boxes highlight the main data points, while the blue squares show additional points used to determine features [14].

2.2 Image segmentation using a pre-trained VGG16 model for Feature Extraction and a Random Forest classifier

Transfer learning is a method where knowledge of neural networks from one task is applied to a different, yet related task. Essentially, it is like taking features that the neural network learned from one set of images and using that knowledge for another set. Many deep learning models, when trained on common images, pick up general features such as edges, shapes, and blobs in the early stages of training that can be useful for various tasks. If the new task has fewer images to learn from, transfer learning helps by using the knowledge from the bigger dataset (ImageNet dataset), preventing the model from making errors due to limited data. In this part, the primary aim was to achieve proficient image segmentation through the combined utility of a pre-trained VGG16 model for feature extraction and a Random Forest classifier for pixel wise classification. VGG16 is a convolutional neural network model developed by K. Simonyan and A. Zisserman applied from the University of Oxford, designed with 16 weighted layers, including 13 convolutional layers [6]. Trained on the extensive ImageNet dataset for image classification, it is recognized for its depth and ability to identify a broad range of objects and features. The model's uniform architecture like in Figure 2, characterized by its consistent use of 3x3 convolutional filters, makes it straightforward to implement and adapt. One way to extract lower-level features from the pictures is to use the VGG16 model for feature extraction in image segmentation, truncating its layers after the second conv2 layer. Simple patterns, textures, and edges are examples of lower level characteristics. The foundation model for the suggested transfer learning method is VGG16, which has been previously trained for object detection tasks using the

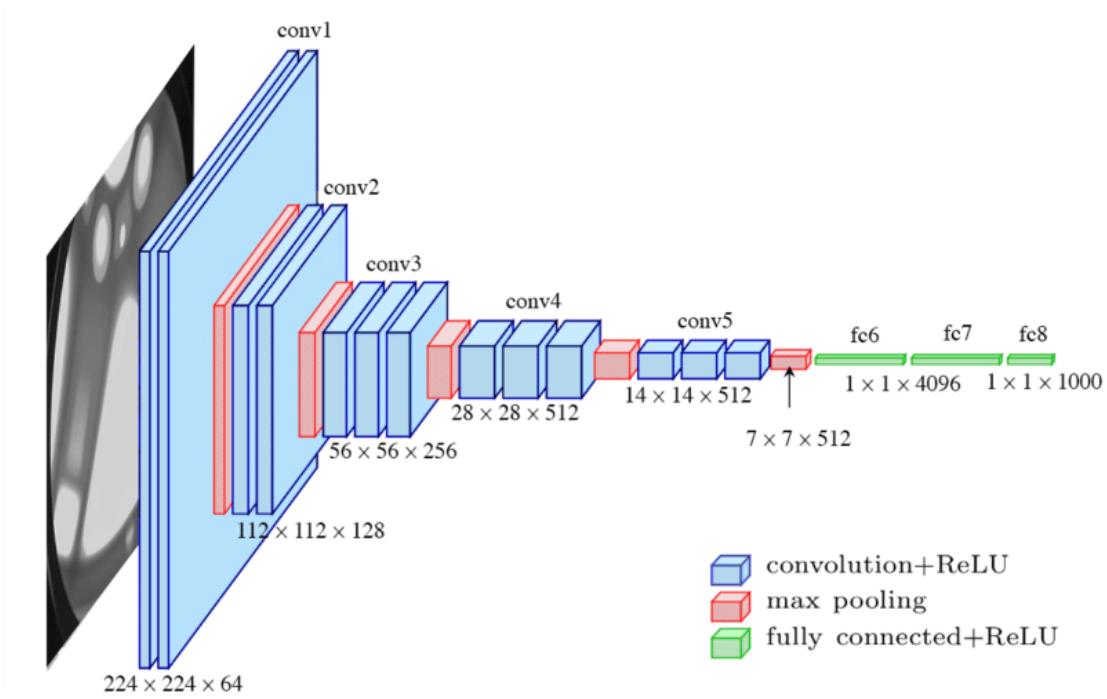


Figure 2: Visualization of the VGG16 architecture [16].

ImageNet dataset. In order to extract the characteristics of pictures that are utilized as input to train a Random Forest classifier, we employ the first two convolutional layers of the VGG16. Next, we swap out the fully connected layers in the suggested model for pixel wise classification, or semantic segmentation, with the trained Random Forest classifier.

We began by importing necessary libraries such as Matplotlib, Numpy, panda, glob, OS, OpenCV, and Keras. In the preprocessing phase, we loaded training images and created a list to store them. These images underwent resizing and color adjustments. Similarly, mask images, which acted as labels, were processed and stored in another list. These lists were then converted into arrays, ready for model training. We then extracted image features Using a pre-trained VGG16 model, ensuring its original weights remained unchanged. To work with a Random Forest classifier, we reshaped the extracted features and labels and eliminated any unlabeled data. After training the classifier, we saved the model for future use. In the testing phase, an external image was prepared and segmented using our trained model, subsequently, the results were visualized and stored [17].

2.3 Semantic Segmentation with U-net

U-Net, introduced by Ronneberger et al. in 2015, is a specialized convolutional neural network design tailored for image segmentation, especially evident in its performance on biomedical images like cell microscopy images. The design of U-Net is characterized by two main sections. The network architecture in Figure 3 refers to the structured design and layout of a neural network. Within such architectures, especially in the context of image processing, there is often a contracting path and an expansive path. The contracting path, typical in a convolutional network (ConvNet), reduces the spatial dimensions of the feature maps while increasing their depth or number of feature channels. As the image size contracts, the network captures more abstract characteristics. This reduction is achieved using operations like 3x3 convolution (a convolution operation with a 3x3 kernel) and 2x2 max pooling (a downsampling method where the maximum value is taken from a 2x2 window). After processing, an activation function, commonly the Rectified Linear Unit (ReLU) which efficiently introduces non-linearity and helps the network learn complex feature representations while avoiding the vanishing gradient problem, is applied. The expansive path, on the other hand, involves upsampling to increase spatial dimensions, which means restoring the feature maps to the original image dimensions. This is often followed by a 2x2 convolution, also known as "up-convolution". To merge features from both paths, concatenation is used, which might require cropping of feature maps to ensure matching dimensions. Lastly, 1x1 convolution is a unique operation used to adjust the depth of feature maps, often seen in the final layers to produce a segmentation map, where each pixel is associated with a class or label. The movement of convolution or pooling operations is determined by the stride, which indicates how many pixels the window shifts during each operation. Backpropagation refers to the process of computing and propagating gradients of the loss with respect to the model's parameters during backpropagation, which is crucial for updating the parameters and training the neural network.1x1 Convolution with Sigmoid Activation is used to reduce or increase the number of feature channels without affecting the spatial dimensions, and when followed by a sigmoid activation function, it produces output values between 0 and 1 for binary segmentation, which can be interpreted as probabilities in a segmentation task [18].

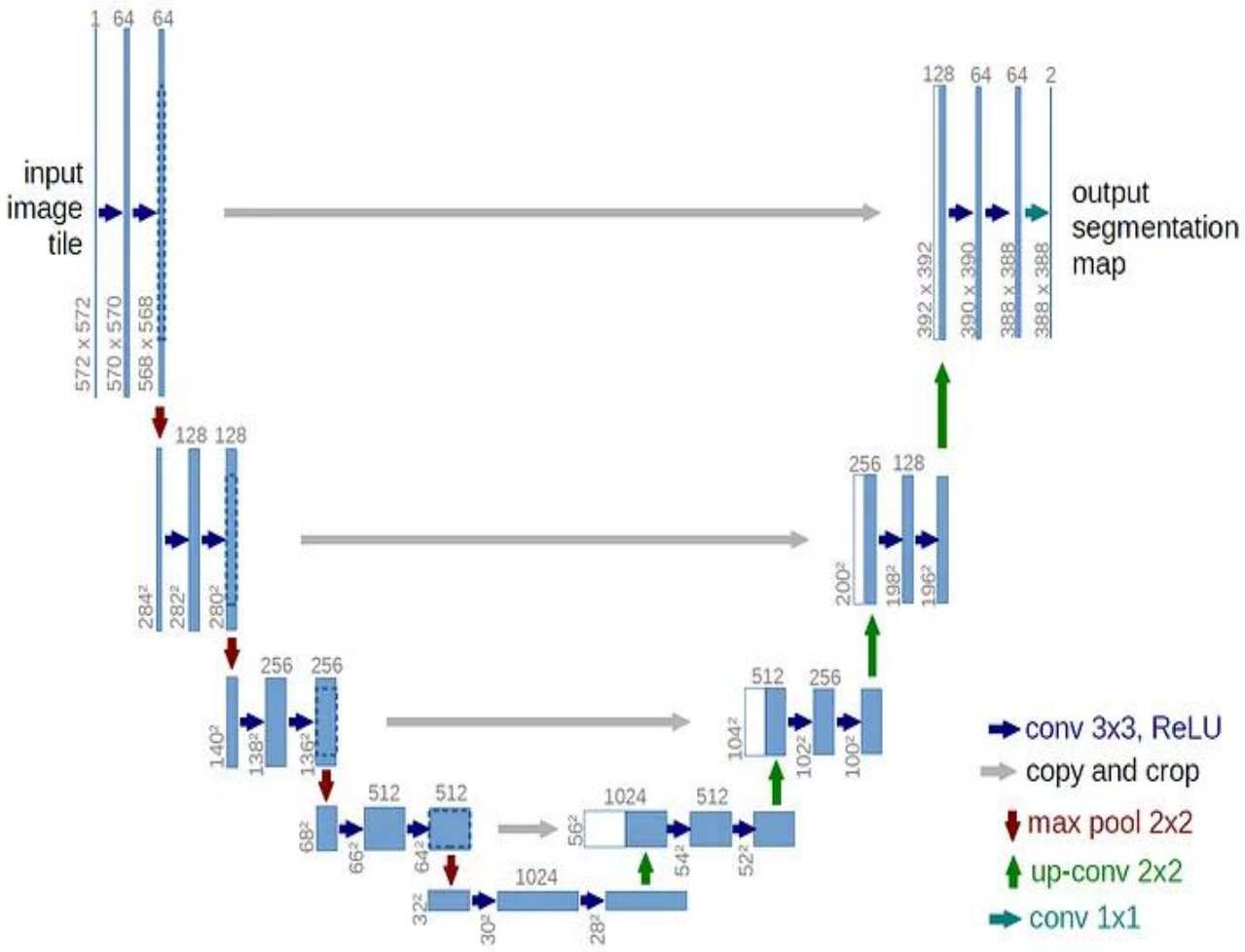


Figure 3: U-net architecture (lowest resolution example for 32x32 pixels). Every blue box represents a feature map with many channels. A number on top of the box indicates how many channels it has. The box's lower left edge has the x-y-size indicated. Copied feature maps are represented by white boxes. The various operations are shown by the arrows.[18]

Initially, libraries such as Matplotlib, Numpy, panda, glob, OS, OpenCV, Keras, Tensorflow, and scikit-learn for image processing and deep learning are imported. The process begins with loading images and their corresponding masks from designated directories, ensuring alignment between the two. These images are converted to a standardized size and undergo grayscale conversion. The pixel values of both the images and masks are then normalized for better training efficiency. The dataset is subsequently split into training and test subsets. Following this, the U-Net Architecture is constructed, which comprises blocks of convolutions in its encoder and decoder sections. The model is then compiled, trained, and saved. Finally, an image from the test set is selected at random, and the model predicts its mask, which is compared against the actual mask for a comprehensive understanding [19].

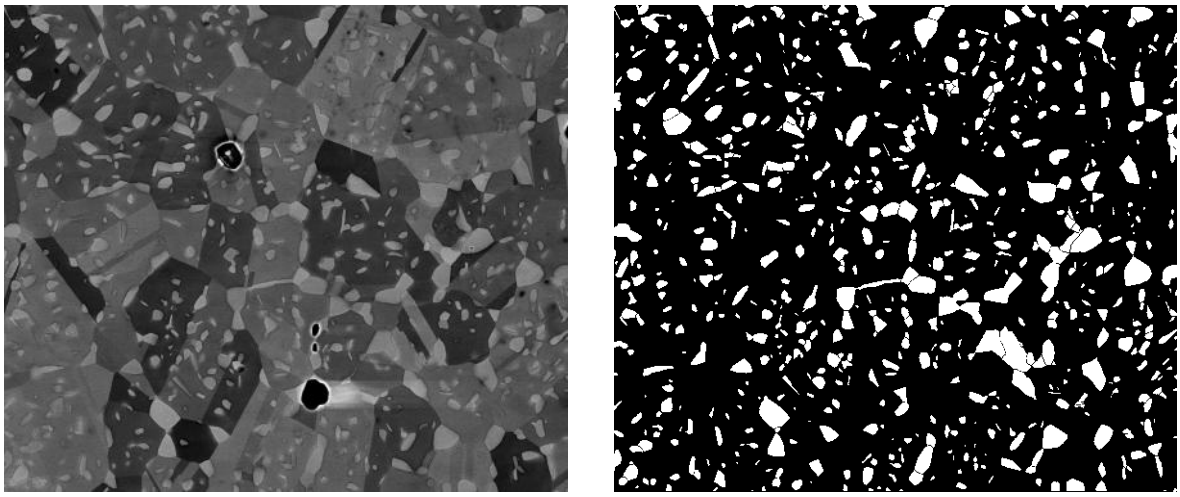
2.4 Semantic Segmentation via Segmentation Models Library

The Segmentation Models library is a top choice for image segmentation tasks due to its user-friendly features. With its high-level API, creating a model for segmentation is simplified to just two lines of code. The library offers four model architectures, including the renowned Unet, and 25 different backbones for each, all equipped with pre-trained weights to ensure better and faster results. These models, many trained on a vast number of images from ImageNet, are adaptable for various tasks, eliminating the need for users to build complex structures like U-Net from scratch. This not only saves time but also reduces potential errors that can arise from manual model building. The library also comes packed with essential tools for segmentation, such as specific losses like Jaccard, Dice, and Focal, and metrics like IoU and F-score to evaluate the models. Its ease of navigation and feature selection make it a wise choice for those aiming for efficiency in their segmentation tasks [20].

Essential libraries such as Numpy, panda, glob, OS, OpenCV, Keras, Tensorflow, scikit-learn, and segmentation models are installed and imported to facilitate the image segmentation process. The backbone architecture for the segmentation model, such as Vgg16 or efficientnetb7, is defined for feature extraction. Images are loaded from a directory, resized, and undergo color conversion, while their corresponding masks are similarly processed. This data is then split into training and validation sets and preprocessed. A U-Net model is established, compiled, and trained for a set number of epochs which refers to one complete forward and backward pass of all the training samples through the model. After training, the model is saved and later retrieved for testing. An external image undergoes segmentation through this model, producing a predicted segmentation mask. This mask is visualized alongside the original image for comparison [21].

3 Results

In our study, we worked with a dataset comprising 12 back scatter electron images of Cr-Mn-Fe-Co-Ni high entropy alloys (HEA), captured using scanning electron microscopy (SEM). The SEM samples photographed came from various conditions, influenced by factors such as annealing temperatures and hours, and the specific areas of the sample being imaged. Each image has dimensions of 2048 pixels by 1776 pixels. Alongside these SEM images, we possess corresponding mask images, which were derived using the Image Segmenter app in MATLAB R2022b. Figure 4 displays a representative SEM image paired with its respective mask image.



a) b)

Figure 4: a) Example of HEA SEM back scatter electron image is annealed at 900 °C for 10 hours. b) Segmented image obtained using Matlab Image Segmenter app.

3.1 Manually Feature Extraction and Random Forest Semantic Segmentation

Our objective is to utilize machine learning techniques to achieve pixel wise segmentation of σ phase particles automatically, rather than relying on manual methods.3 different models were trained in this approach can be analyzed in Table 1. Initially, to harness the potential of machine learning for this task, we need image features. These features are derived from the application of various filters, including Gabor, Canny edge, Robert, Sobel, Scharr, Prewit, Median, and Variance, to a 2048x1776 pixel training image, as depicted in Figure 4a. Once these features are extracted, they are stored into a data frame, along with the mask pixel values that serve as labels and the pixel values of the original training image. In the mask image, there are two distinct classes: the background, with pixel values of 0, and the foreground, representing σ phase particles, with pixel values of 255. To train and evaluate our machine learning model, we partitioned the dataset into training and test subsets. For Model 1, we employed a Random Forest classifier with empirically chosen 50 estimators for the training process. The evaluation of the model on the test dataset yielded a prediction accuracy of 95.91%. A detailed breakdown of the importance values of the filters used to generate

the feature vector obtained from the sci-kit-learn library feature importances function is presented in Figure 5.

Table 1: Comparison of models details where Random Forest is used as a classifier to predict foreground and background.

Model Name	**Classifier Name**	**Images**	**Size**	**Number of Estimators**
Model 1	Random Forest	1	2048x1776	50
Model 2	Random Forest	5	2048x1776	100
Model 3	Random Forest	100	224x224	100

After training, we applied our model to a previously unseen image derived from differently-treated metal. The goal was to determine the model's ability to identify σ phase particles in this new context. The original image and the model's prediction are presented in Figure 6.

We utilized a second method for Model 2 where our dataset comprised five training images of 2048x1776 pixel resolution, along with their matching masks. During the training phase of our machine-learning model, these images underwent various filter applications. We created 32 separate Gabor filters and With all filters included, our feature dataset has a total of 41 columns. We partitioned the feature dataset and mask pixel values into training and test sets. The model was then trained using a Random Forest classifier with empirically chosen 100 estimators. The test dataset yielded an accuracy of 95.79%. This trained model was further used on a distinct test image, and the outcomes are presented in Figure 7.

In the third approach, we increased our training dataset by subdividing the original five training and mask images (each 2048x1776 pixels) into 100 smaller training and mask images, each image has the size of 224x224 pixels according to Vgg16 architecture shown in Figure 2. Following the same feature extraction methodology, we split the data into training and test sets. We used a Random Forest classifier with 100 estimators for training. The training time was reduced to 40 minutes, in contrast to around 2 hours in our previous method. The results from this trained model on a different test image are shown in Figure 8. This figure displays the original image, the manually segmented version, and the prediction from Model 3, facilitating a direct comparison between them.

In Figure 8c, several regions are marked for further discussion. The σ particle detected only its edge is surrounded by a blue circle. An area with uncertain σ phase particle predictions is marked with a green rectangle. The σ particle that closely matches the prediction is encased in a red circle.

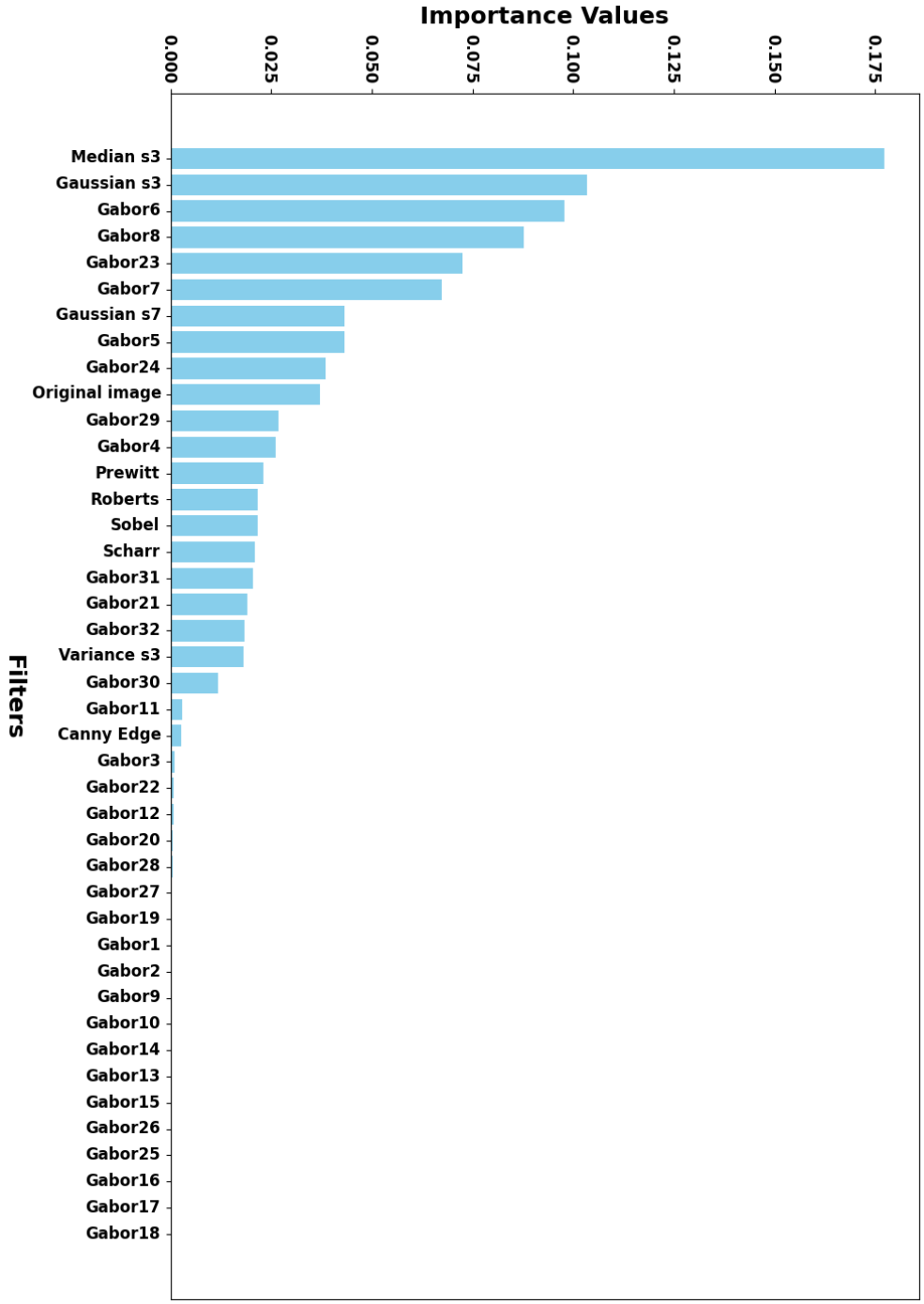
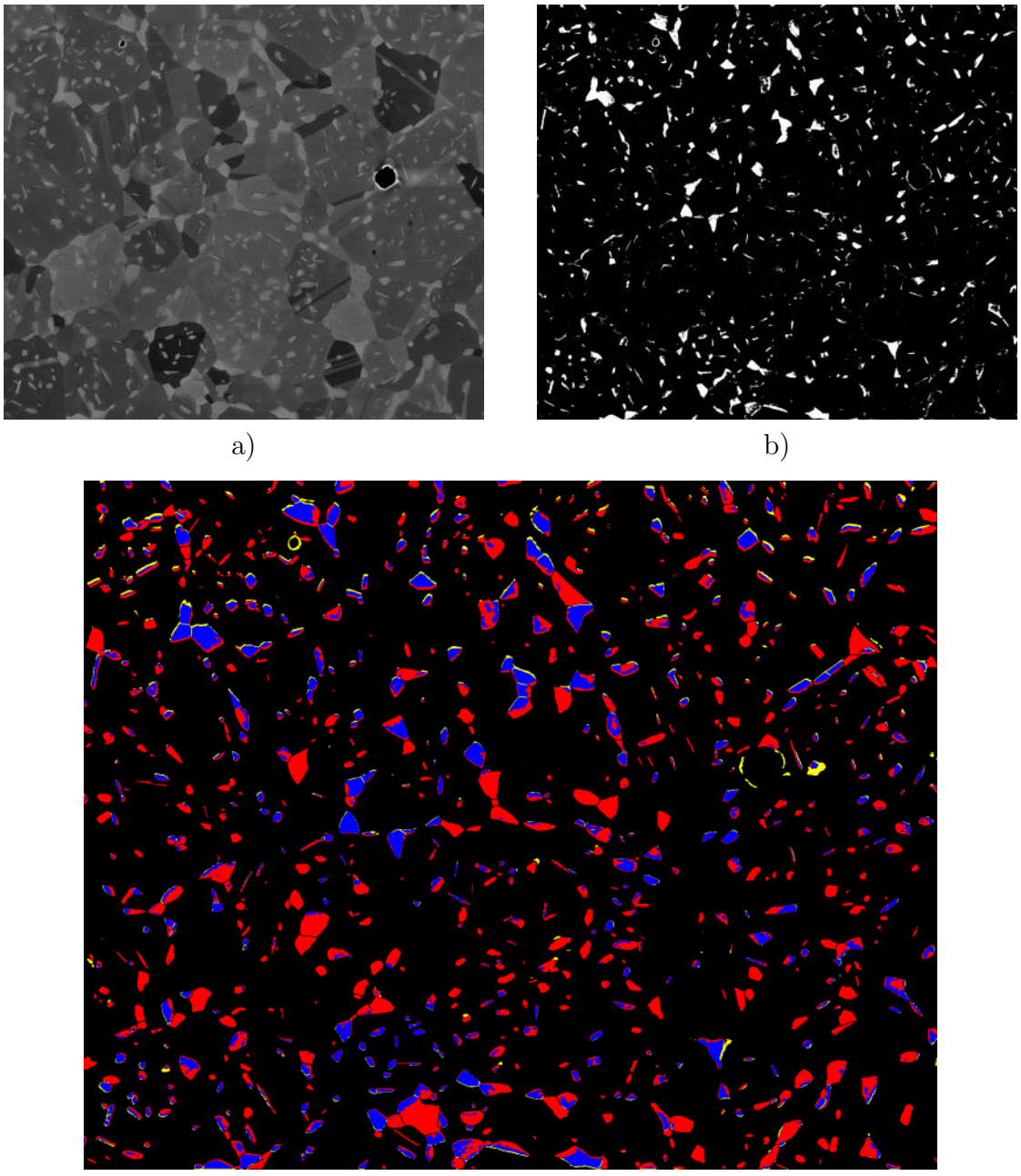


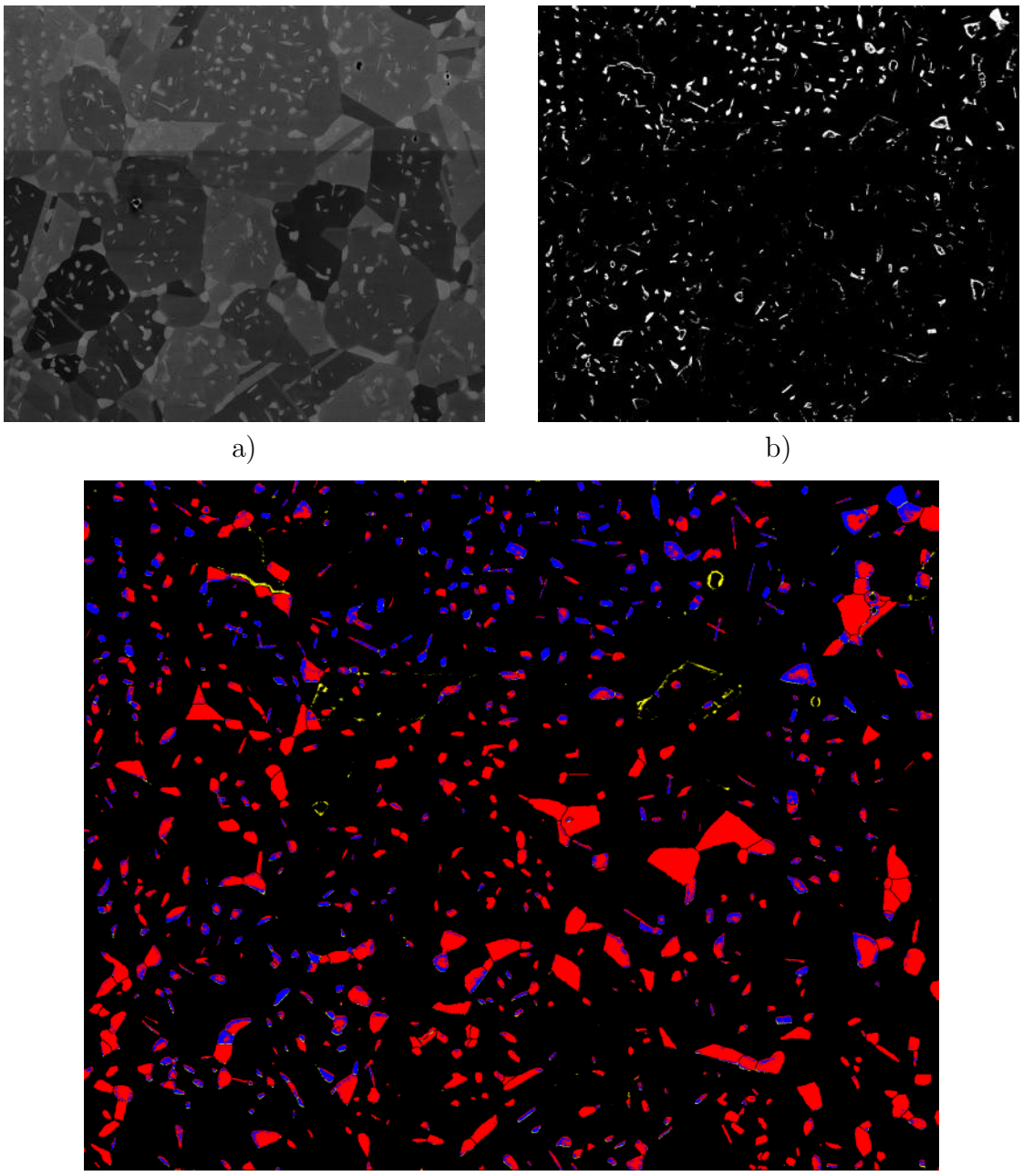
Figure 5: Feature importance values of filters ranked from most to least important for Model 1.



a) b)

c)

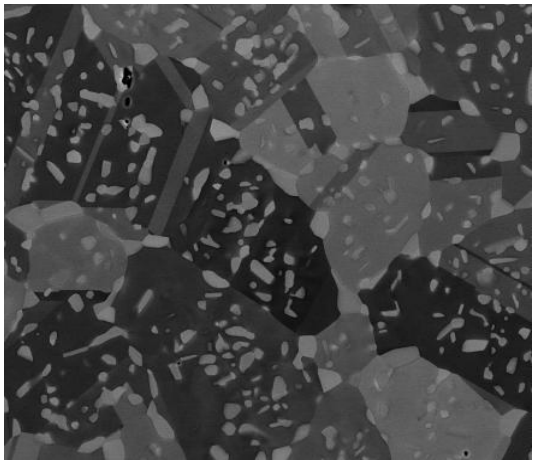
Figure 6: a) Test image with different morphology and processing conditions. b) Pixel wise segmented image by Model 1. c) The overlapped image distinguishes the manually segmented image with the predictions from Model 1. Blue regions represent overlaps, red highlights areas unique to manual segmentation, and yellow areas unique to predictions from Model 1.



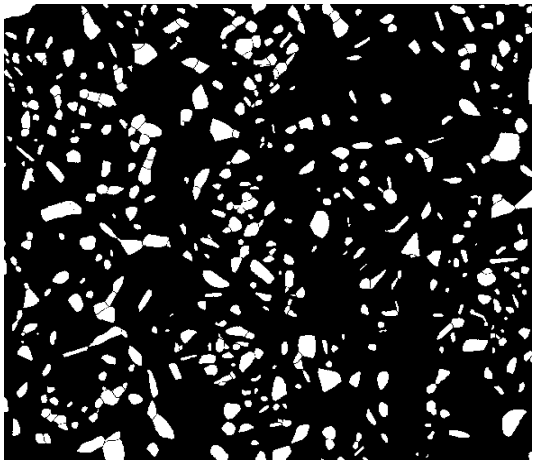
a) b)

c)

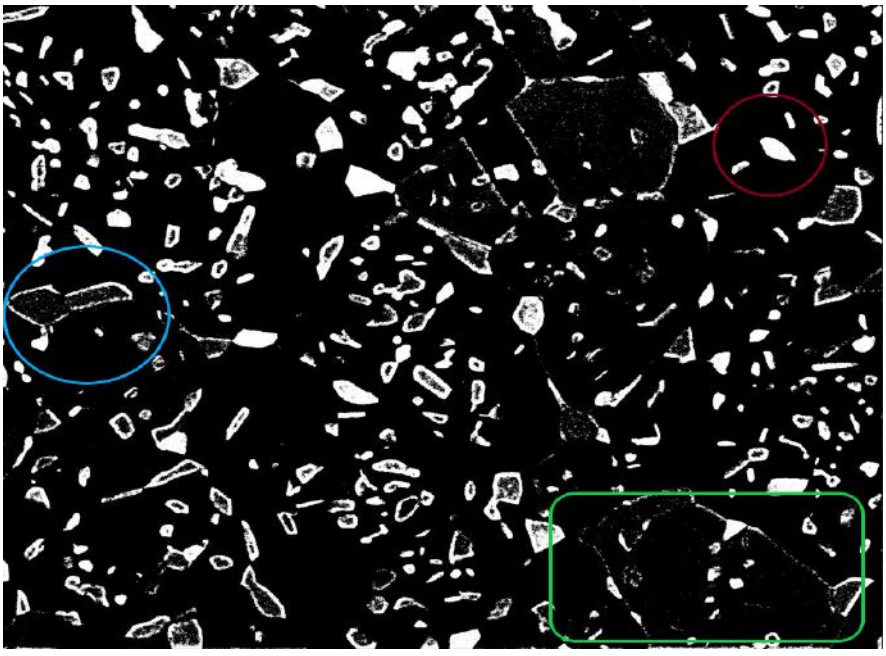
Figure 7: Test image with different morphology and processing conditions. b) Pixel wise segmented image by Model 2 c) The overlapped image distinguishes the manually segmented image with the predictions from Model 2. Blue regions represent overlaps, red highlights areas unique to manual segmentation, and yellow areas unique to predictions from Model 2.



a)



b)



c)

Figure 8: a)Test image with different morphology and processing conditions. b) Manually segmented image using Matlab image segmented app. c) Pixel wise segmented image via Model 3, trained on 100 image patches. Red circles indicate complete particle segmentation, blue circles show partial σ particle edges and green circles highlight overestimations and accurate predictions.

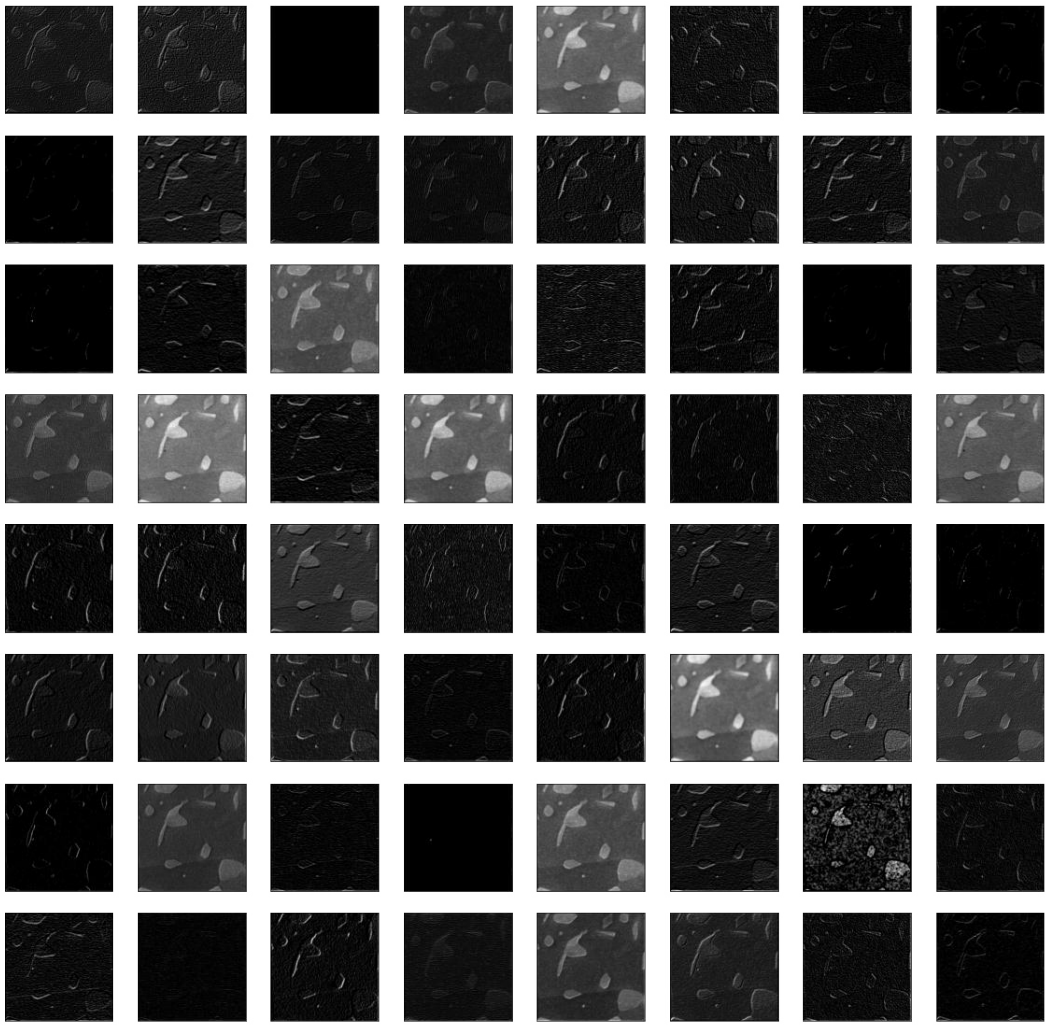
3.2 Image segmentation using a pre-trained VGG16 model for Feature Extraction and a Random Forest classifier

In our fourth approach, a slightly different approach was considered, exploring whether classic machine learning algorithms, combined with Transfer Learning, could effectively accomplish the task. Two models were trained as illustrated in Table 2. Initially, the VGG16 convolutional neural network, designed for classification problems, was sourced from the Keras library in Python. The first two layers of this network were leveraged for feature extraction, given the consistent dimensions in these initial convolutional layers. For RF1 model, 50 out of the 100 previously segmented images, each image has the size of 224x224 pixels according to Vgg16 architecture shown in Figure 2, were utilized.

Table 2: Comparison of models details where vgg16 is used as a feature extractor.

Model Name	**Classifier Name**	**Images**	**Size**	**Number of Estimators**
Model RF1	Random Forest	50	224x224	50
Model RF2	Random Forest	20	224x224	20

When these images were processed through the VGG16 model using its pre-trained weights, 64 features were extracted for each image due to the first 2 convolutional layers of Vgg16 being used, illustrated in Figure 2. Applied 64 convolutional filters to a patched image and the most important 3 filters obtained using feature importances function from scikit-learn library are shown in Figure 9.
By pairing these features with the mask pixel values of the 50 images, a dataset was compiled for training purposes. The model, referred to as RF1, was trained using the Random Forest classifier with 50 estimators, and 50 images, and subsequently tested on a test image. In a parallel effort, another model, RF2, was trained using 20 of the segmented images (224x224 pixels) with 20 estimators, maintaining the same feature extraction method, and was also tested on the same test image. Specifications of the two models are illustrated in Table 2. For a clear side-by-side analysis, the results from both models are displayed in Figure 10 and Figure 11.In the predicted images, certain σ phase particles appear incompletely detected. Although their shapes are distinguished, observable in the area enclosed by a green rectangle. Conversely, there are regions with excessive predictions, as indicated by the section enclosed in a red rectangle.



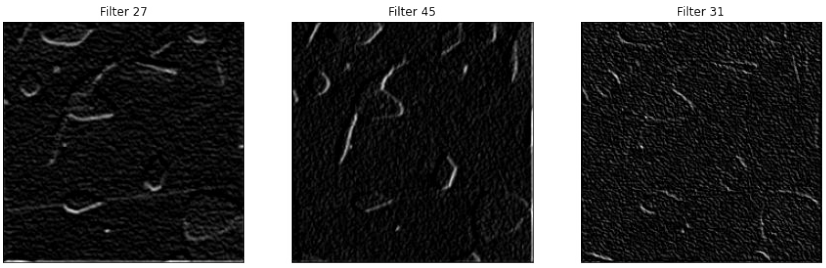


Figure 9: A patched image was processed through the first two convolutional layers of Vgg16, resulting in 64 features. The top three filters out of these 64 were identified using the feature importance function from the Python sci-kit-learn library.

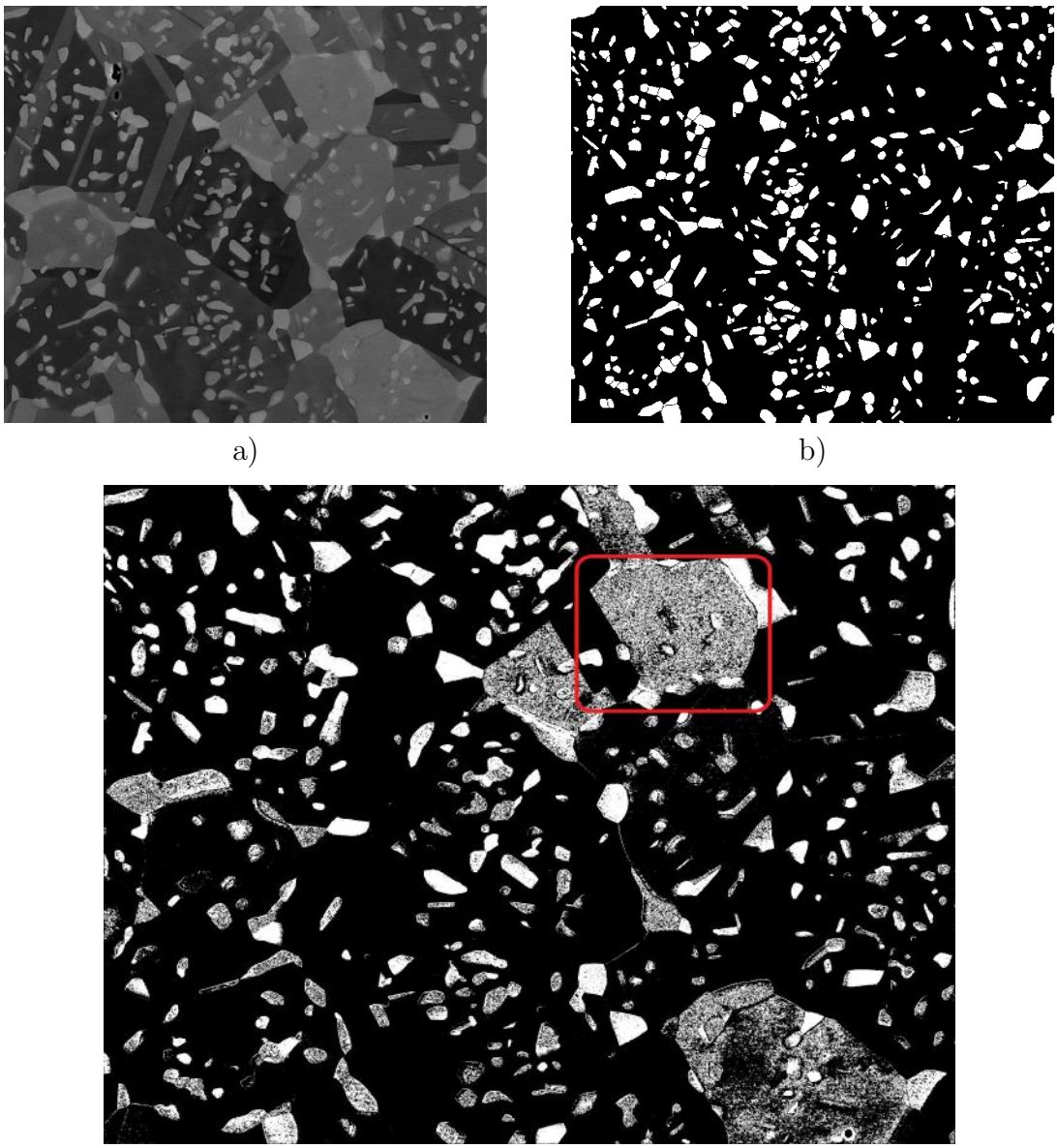
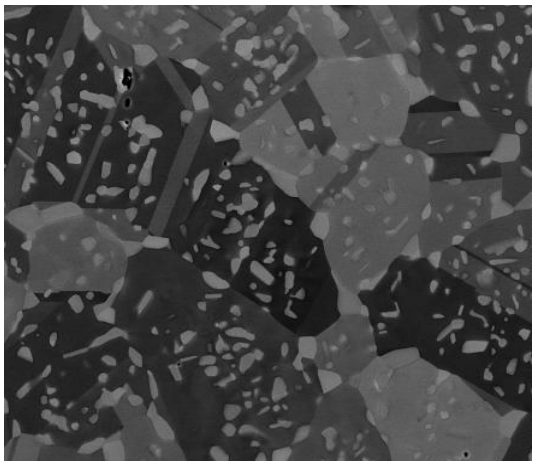
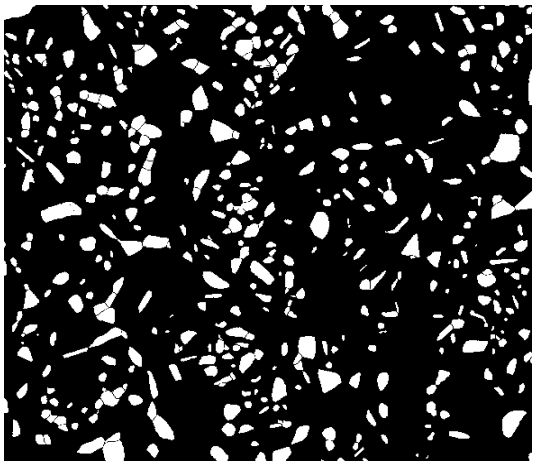


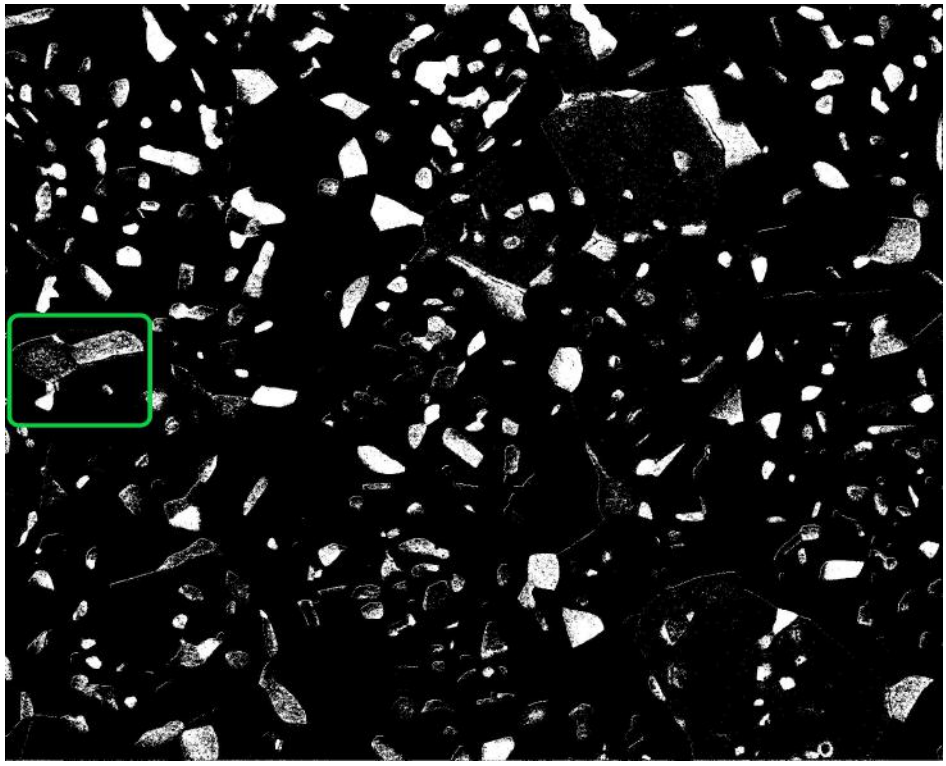
Figure 10: a)SEM image. b) Manually segmented image using Matlab image segmented app. c) Pixel wise segmented image by an RF1 model trained with patched 50 images (224x224 pixels) and masks. The red circle indicates the representative area where overprediction occurred.



a)



b)



c)

Figure 11: a) SEM image. b) Manually segmented image using Matlab image segmented app. c) Pixel wise segmented image by an RF2 model trained with patched 20 images (224x224 pixels) and masks. The green rectangle indicates the representative area where underprediction occurred.

3.3 Semantic Segmentation with U-net

In the methods section, we described the workings of the U-net. For the fifth approach, we trained three U-net models using grayscale images and masks, sized at 128x128 and 256x256 pixels. Details are illustrated in Table 3.The first model U-net 1, trained with 1000 images, utilized 75 epochs and a learning rate of 0.01 as hyperparameters. The second model U-net 2, trained with 1500 images, also ran for 75 epochs but with a slightly reduced learning rate of 0.001. After training U-net 2 model, we plotted the loss and accuracy metrics for both the training and validation datasets against the number of epochs. These results are illustrated in the Figure 12. The loss function reached its lowest value, and the accuracy varied between 0,92 and 0,96.

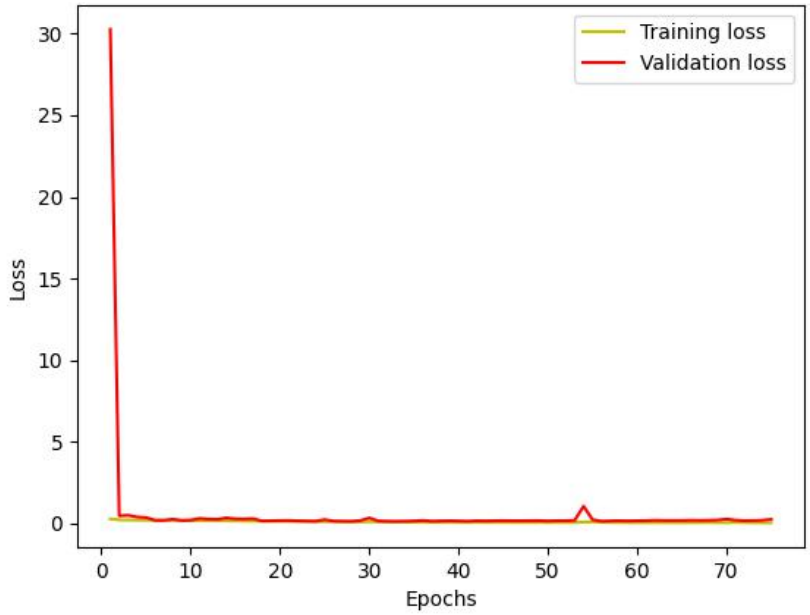
Table 3: Comparison of U-net models and manual segmentation.

Model	**Images**	**Size**	**Epochs**	**Learning Rate**	**σ labels (%)**
Segmentation via Matlab	-	-	-	-	15.14
U-net 1	768	128x128	75	0.01	11.91
U-net 2	768	128x128	75	0.001	12.62
U-net 3	768	256x256	50	0.001	17.03

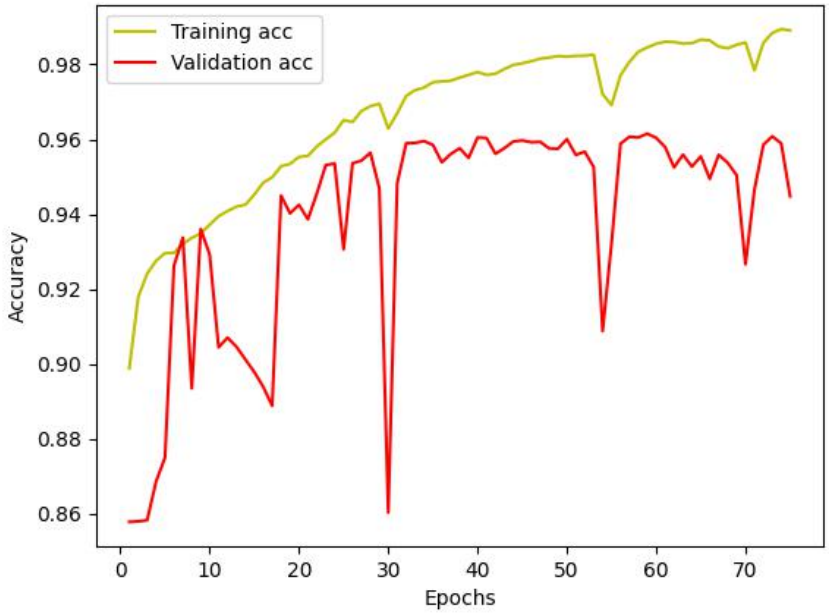
The outputs of the two models are presented in Figure 13 and Figure 14. Areas marked in red and purple in Figure 13c and in Figure 14c display the detected σ phase particles by the models, whereas these particles are absent in the manually segmented image in Figures 13b, 14b. In Figure 13c, σ phase particles are only partially identified, highlighted by the green circle. Meanwhile, in Figure 14c, the U-net 2 model detects these particles, though not fully, they retain their shape, as highlighted by the turquoise circle. Still, in the manually segmented image (Figure 13b), these particles are not present in that region.

Furthermore, we trained the third model U-net 3 using a dataset consisting of 936 images and corresponding masks, each with a size of 256x256 pixels. The training process involved 50 epochs, and we utilized a learning rate of 0.001 as a hyperparameter. After training the third model, a graph showing the loss and accuracy metrics for both the training and validation datasets was produced, as seen in Figure 15. The loss function reached a minimum value, while the accuracy varied between 0,88 and 0,94.

The outcomes of the U-net 3 model were compared to manually segmented image is depicted in Figure 16. In Figure 16c, the yellow areas highlight the predictions of the third model for σ phases. These predictions can be compared to the manually segmented images and the predictions from the U-net 2 model. In Figure 16c, the U-net 3 model predictions highlight σ particles within the red circles, while the second model's predictions differ in these regions. In the areas marked by red circles, some σ particles are not present when compared to the manually segmented image. Table 3 provides a comparison of σ phase percentages between the images segmented by the three models and the manually segmented image.

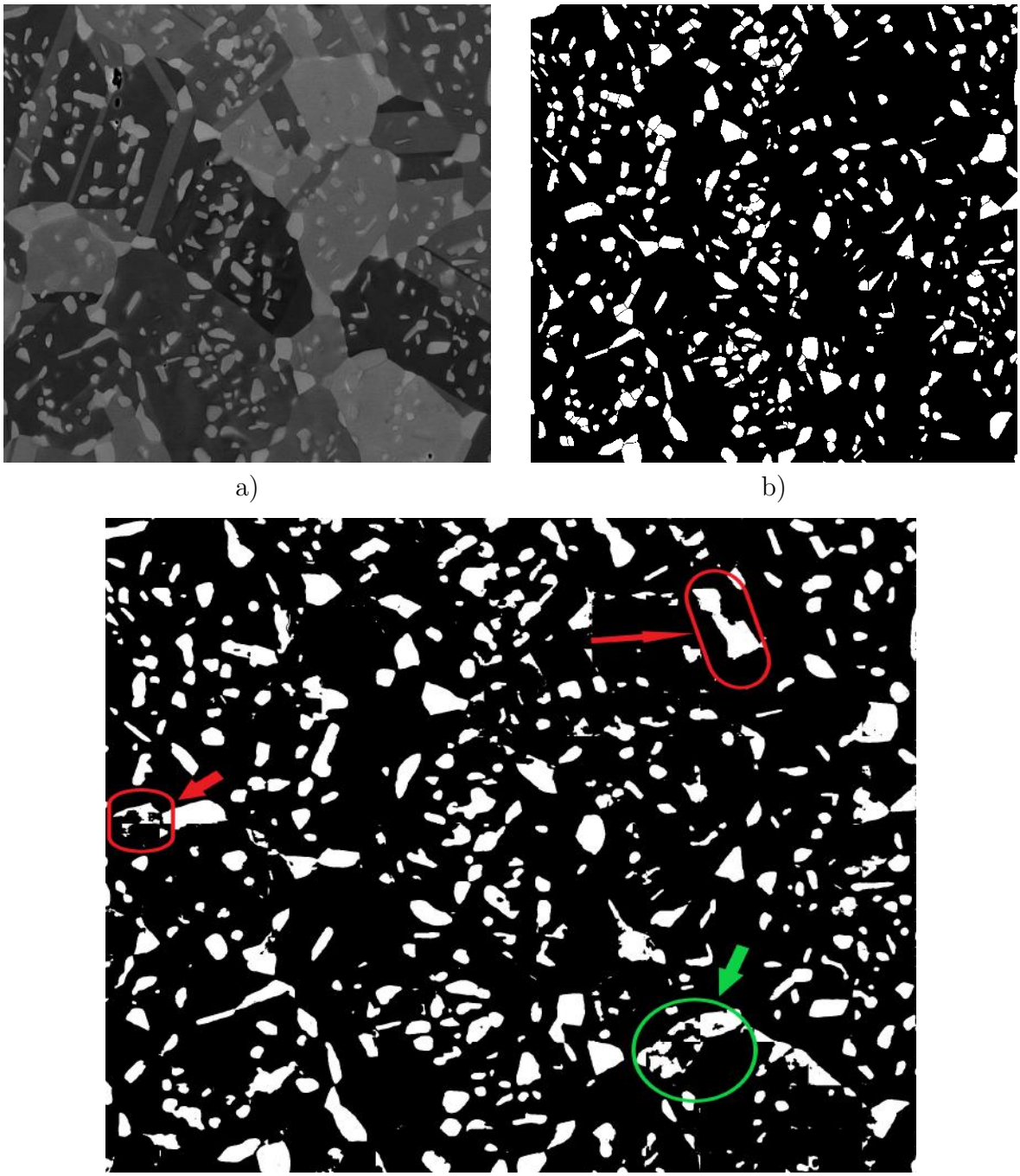


a)



b)

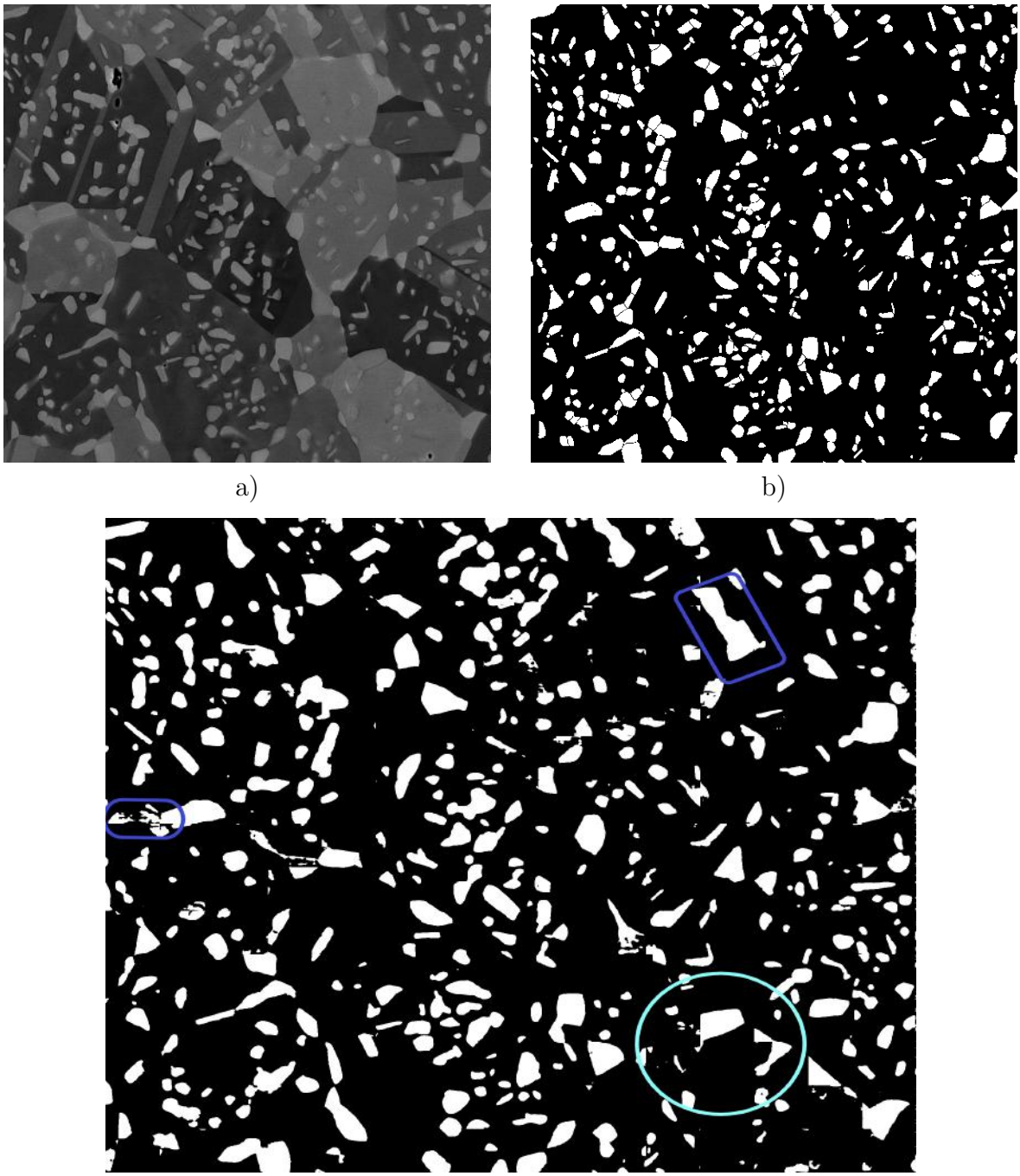
Figure 12: a)Loss metrics on training and validation dataset of the U-net 2 model trained with 1500 128x128 pixels images and masks. b) Accuracy metrics of the U-net 2 model trained with 1500 128x128 pixels images and masks.



a) b)

c)

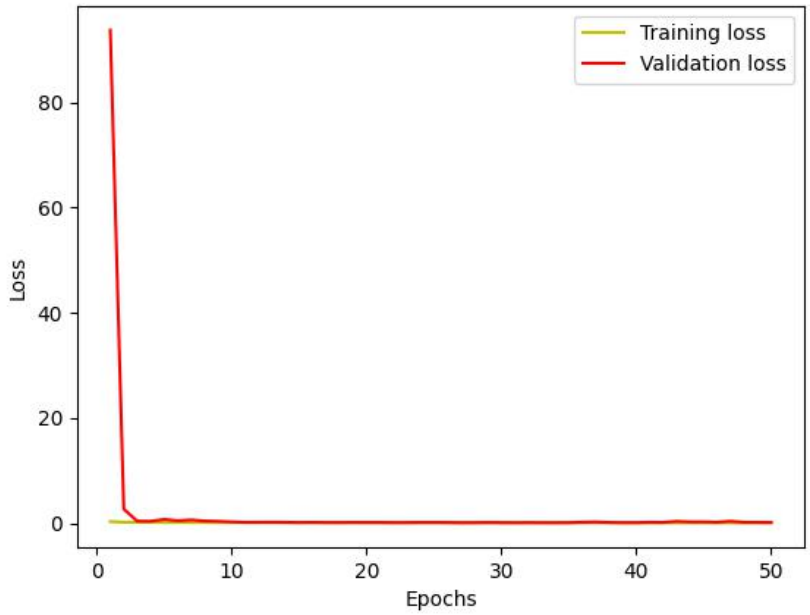
Figure 13: a) Test image obtained via SEM b)Manually segmented image c) Predicted segmentation via U-net 1 model using 128x128 pixels 1000 masks and original SEM images.



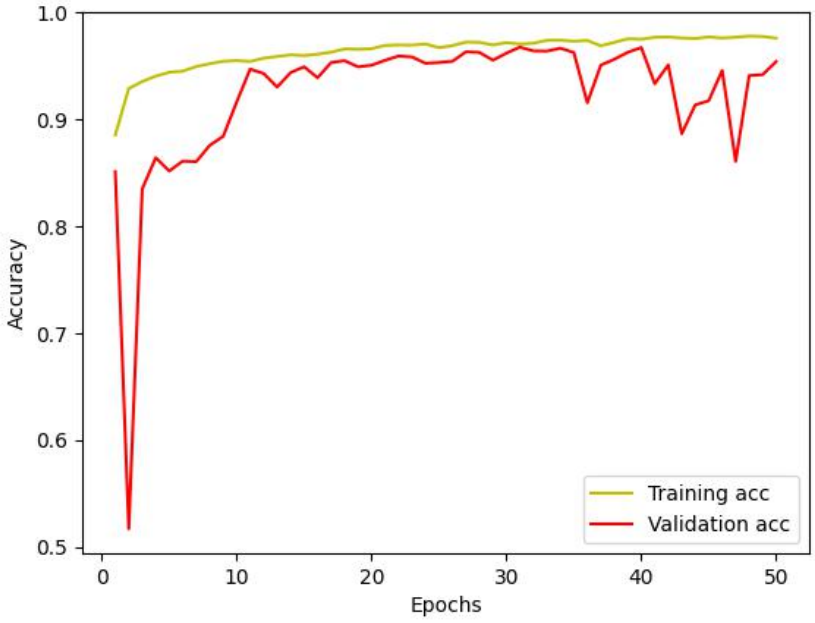
a) b)

c)

Figure 14: a) Test image obtained via SEM b)Manually segmented image c)Predicted segmentation via third U-net 2 model using 128X128 pixels 1500 mask and original SEM images.

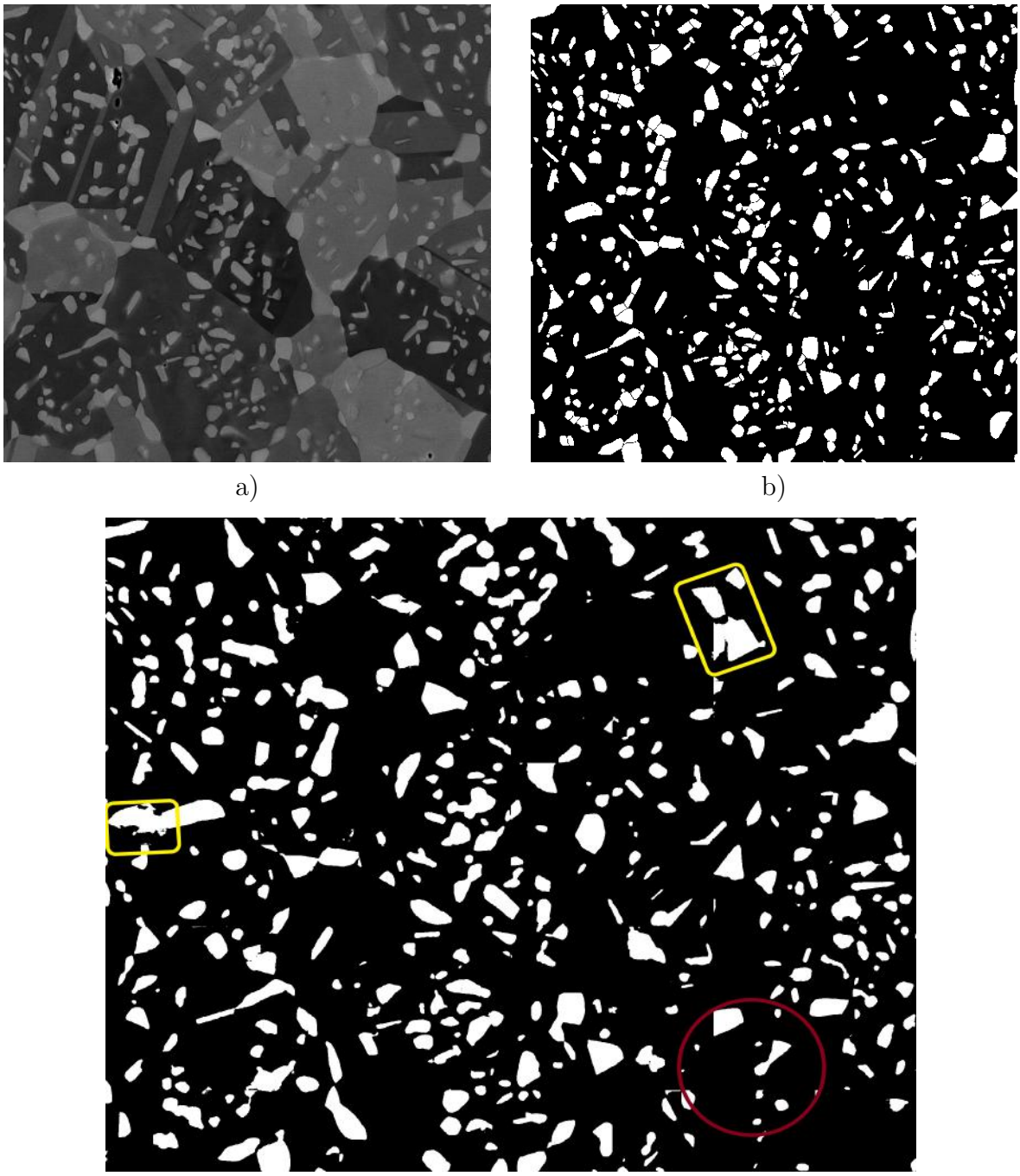


a)



b)

Figure 15: a)Loss metrics of the model U-net 3 on training and validation dataset b) Accuracy metrics of the U-net 3 model



a) b)

c)

Figure 16: a) Test image obtained via SEM b)Manually segmented image c)Predicted segmentation via model U-net 3 using 256x256 pixels 936 mask and original SEM images.

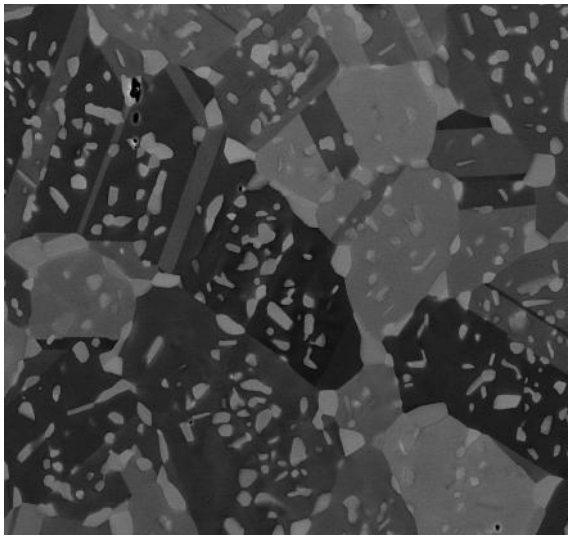
3.4 Semantic Segmentation via Segmentation Models Library

In our last approach, we used the U-net deep learning structure and added the benefit of transfer learning. We started by creating three models: the Vgg16 model used the Vgg16 backbone, the InceptionResNetV2 model used the InceptionResNetV2 backbone, and lastly EfficientNetB7 model used the EfficientNetB7 backbone from the segmentation models library. Details are illustrated in Table 4. These models were trained in 50 epochs. They use pre-trained weights from the ImageNet dataset to help identify features in our training data. Our training set for these 3 models consisted of 768 SEM images (256x256 pixels) and their corresponding masks. The results from these trained models can be seen in Figure 17, Figure 18, and Figure 19.

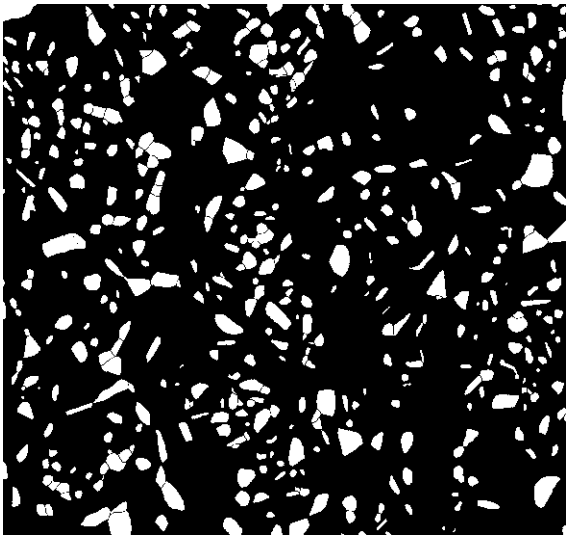
Table 4: Comparison of σ phase percentage obtained from segmented images via three U-net& Transfer Learning models

Model	**Images**	**Size**	**Epochs**	**σ phase labels (%)**
Segmentation via Matlab	-	-		15.14
Vgg16	768	256x256	50	11.38
InceptionResNetV2	768	256x256	50	12.02
EfficientNetB7	768	256x256	50	12.31

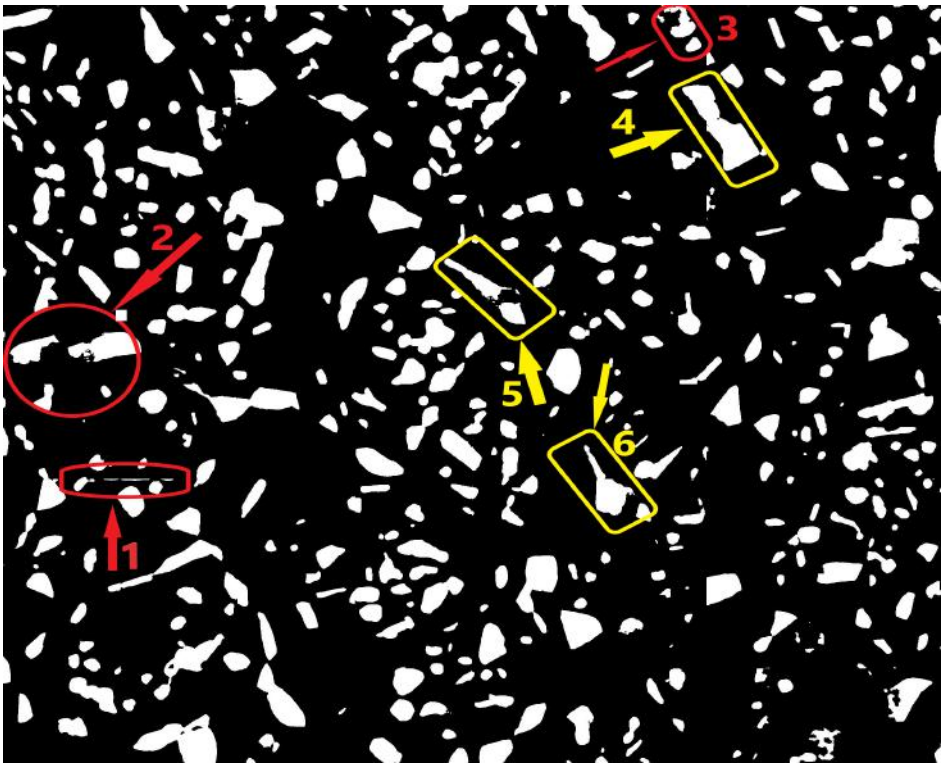
In Figure 17c, the areas pointed out by red arrows 2 and 3, and yellow arrows 5 and 6, show σ phase particles that aren't fully detected when compared to the original in Figure 17a. The location marked by red arrow 1 shows places where the model made incorrect predictions. Notably, the region indicated by yellow arrow 4 highlights a σ phase particle not spotted in the manually segmented image in Figure 17b. This region aligns with the original image, indicating the model's correct detection.In Figure 18c, the areas pinpointed by red arrows 2 and 3 and yellow arrows 5, 6, and 7 show improved predictions compared to Figure 17c. There is an incorrect prediction highlighted by red arrow 1 compared to the original and manually segmented image. Also, zones marked by yellow arrows 4 and 8 indicate an over-prediction of σ phase particles that weren't detected in the manually segmented image from Figure 18b. The comparison of σ particle percentages detected by the models and those in the manually segmented image is presented in Table 4. In Figure 19c, the model uses the EfficientNetB7 backbone for feature extraction and yields predictions that closely match the original image. The σ phase particle shapes in this predicted image correspond closely to those in the original image from Figure 19a. The U-net model integrated with Transfer Learning, utilizing the EfficientNetB7 backbone, stands out as our most effective model to date as the result is illustrated in Figure 19c where the manual segmentation align closely, the model refines the segmentation by eliminating undetected σ particles overlooked by a human expert. The evaluation metrics for this model are detailed in Figure 20. The data shows that both the Loss and Mean Squared Error (MSE) are at their lowest values, while the Intersection over Union (IoU) score and the F1-score are at their highest. We subjected our model to a new test image with different morphological characteristics than the previous ones. This image predominantly showcases needle like σ particles in contrast to the earlier rounded ones. The segmentation outcome using the EfficientNetB7 model is depicted in Figure 21.In certain areas, as indicated by the red circle, the model's predictions align well with the σ phase particles in the original image shown in Figure 21.



a)

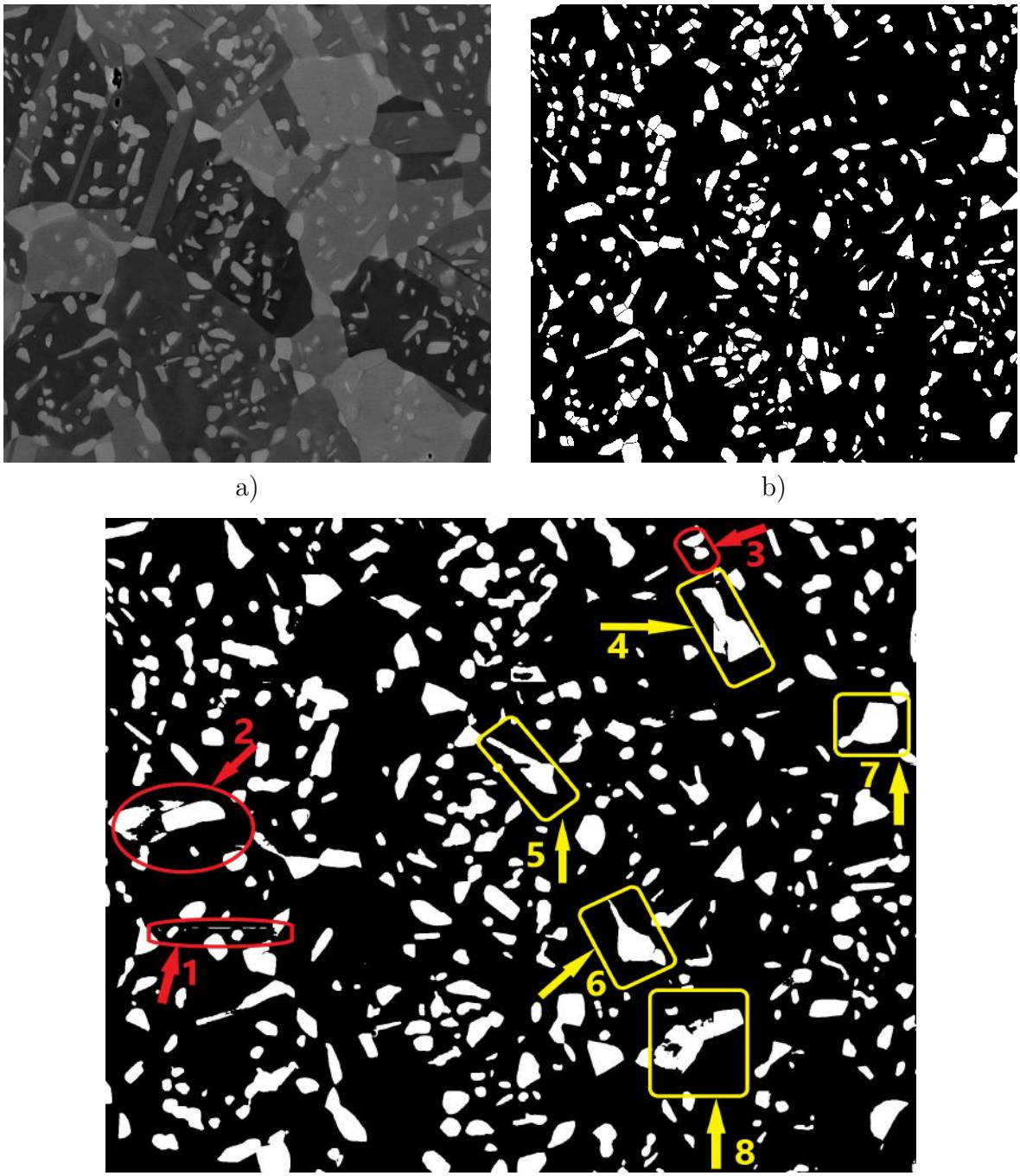


b)



c)

Figure 17: a) Test image obtained via SEM b) Manually segmented image c) Predicted segmentation via the Vgg16 model in which the Vgg16 backbone is used.



a) b)

c)

Figure 18: a) Test image obtained via SEM b) Manually segmented image c)Predicted segmentation using the InceptionResNetV2 model in which the InceptionResNetV2 backbone is used.

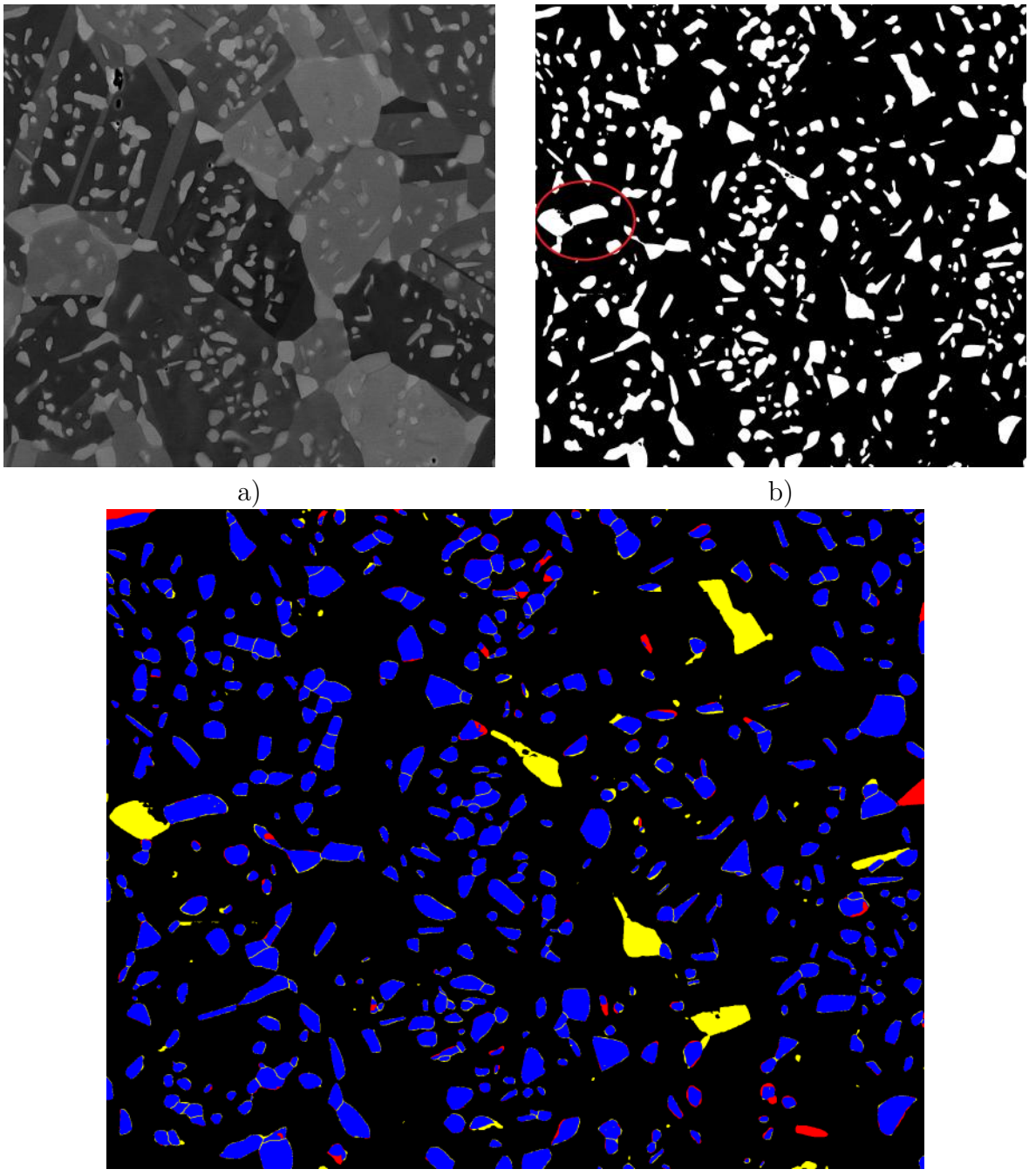


Figure 19: a)Test image obtained via SEM b) Predicted segmentation using the EfficientNetB7 model in which EfficientNetB7 backbone is used. The overlapped image distinguishes the manually segmented image with the predictions from Model EfficientNetB7. Blue regions represent overlaps, red highlights areas unique to manual segmentation, and yellow areas unique to predictions from Model EfficientNetB7.

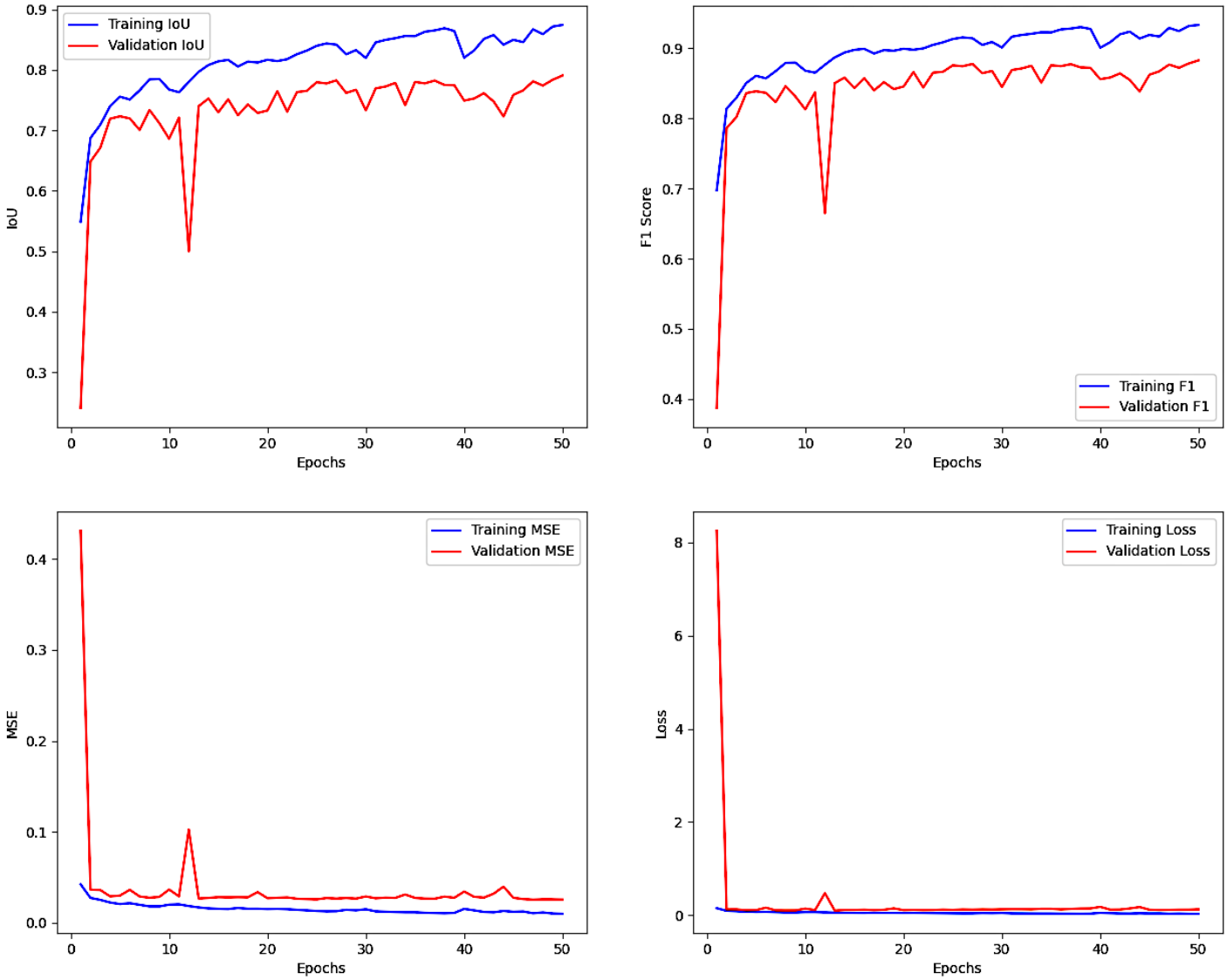
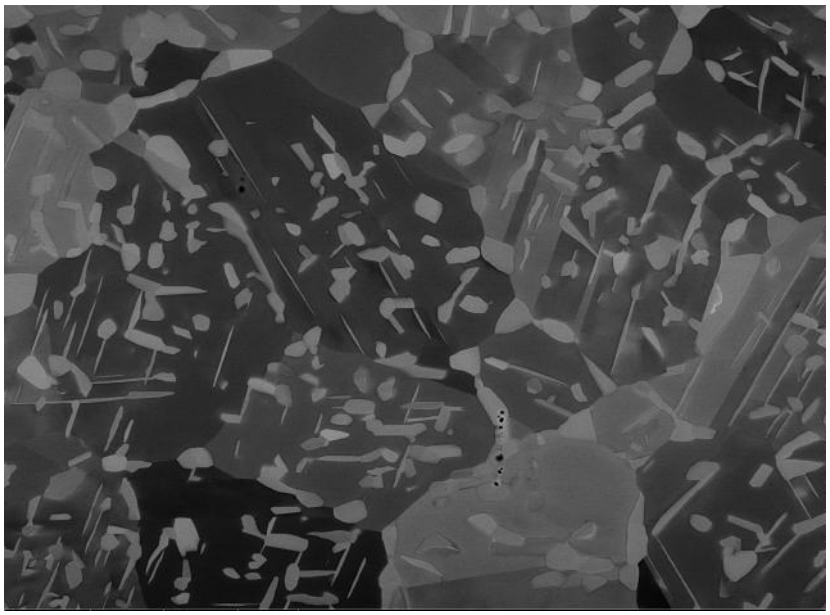


Figure 20: Model evaluation metrics of the EfficientNetB7 model which is trained with EfficientNetB7 backbone.



a)

b)

Figure 21: a)Testing image where it has the different morphological structure of σ phases. b) Predicted segmentation of new test image using the model in which EfficientNetB7 backbone is used.

4 Discussion

Manually Feature Extraction and Random Forest Classifier

In this section, a unique approach and relatively classical machine learning is explored to segment images. In supervised image segmentation, both original images and their corresponding mask images are required. Features must be extracted from the original images. For pixel wise image segmentation, a feature vector that describes the image's characteristics must be created. This vector is then compared to ground truth images using a classifier known as Random Forest. By doing this, characteristics like the size of particles, edges, blobs, patterns and textures are learned by the Random Forest classifier.For the Model 1, the training dataset comprised an original image of dimensions 2048 x 1776 pixels, its corresponding mask image, and a set of extracted features. Traditional filters, such as Gabor, Canny edge, Sobel, and Prewitt, were employed on the original image, leading to the extraction of features, as illustrated in Figure 5. By comparing these features with the ground truth values from the mask image, the weights of 50 random forest estimators were optimized during the training process. When the model, trained on this dataset, was applied to a different large image of the same dimensions, the resulting predictions showcased in Figure 6 demonstrated the potential of machine learning in segmenting σ phase particles. Notably, several σ phase particles were successfully segmented in Figure 6.

After observing positive results from the Model 1, it was believed that adding more images to the dataset might enhance the segmentation. Model 2 was trained using five large training images (2048x1776 pixels) and their corresponding masks with the same first model parameters. However, as shown in Figure 7, increasing the number of images did not lead to better segmentation results. Additionally, the training process was time-consuming due to the computer's Central Processing Unit (CPU) having limited power, taking approximately 4 hours. This extended duration can be attributed to the large image sizes (2048x1776 pixels), which were challenging for the processors to handle.

Large training images and their corresponding masks were divided into smaller-sized images, with the goal of reducing training time. By dividing these large images into small patches, the dataset was expanded, potentially leading to improved segmentation predictions. This is because the classifier could learn patterns from a greater number of examples. The five large training images and masks were divided into 224x224 pixel images, resulting in a total of 100 images and masks.Model 3 were then trained using 100 random forest classifier estimators and 100 images. As illustrated in Figure 8, the segmentation quality improved significantly with reduced noise and clearer σ particle shapes. Training time was also cut down considerably to 40 minutes. In conclusion, while the number and size of images play a role in achieving good results, the number of features extracted from the original training images is crucial for effective segmentation.

Feature extraction by Pre-Trained Vgg16 and Random Forest Classifier

The number of features was increased by utilizing weights from the pre-trained VGG16 deep convolutional neural network classification algorithm. VGG16, trained on the ImageNet dataset for classification, has an accuracy of approximately 93%. Observing the VGG16 architecture, it is evident that the image size varies across convolutional layers. To compensate this size variation issue, the initial two layers were removed. This adaptation of the VGG16, with its pre-trained weights, provided an additional 64 features for each training image, having more features than the previously used method. Subsequently, the Random Forest algorithm was employed for classifi-

cation, yielding significantly improved segmentation results. These outcomes can be viewed in Figure 10 and Figure 11. The RF1 model was trained using 50 estimators on 50 images and masks, each of size 224x224 pixels. In contrast, the RF2 model utilized 20 estimators on 20 images of the same size. The results in Figure 10 and Figure 11 suggest that increasing both the number of images and estimators can lead to overestimation while using fewer images predictions can result in low quality. These results underscore the importance of both the quantity and type of features. The initial layers of a Vgg16 CNN typically capture basic elements like edges and blobs. To have more complex features such as patterns and textures, deeper neural networks are essential for feature extraction.

Semantic Segmentation with U-net

U-Net is a deep-learning architecture designed for image segmentation. It first compresses the input image through a series of layers to capture essential features and then expands it, using these features, to generate a detailed segmentation map. This structure allows U-Net to consider both fine details and broader context within the image. Indeed, U-Net has the potential to offer improved segmentation results for the images that were previously evaluated with other models. Three U-net models were developed for semantic segmentation using U-net. The training was done on Google Colab's GPU to manage the intensive computational demands. The specifics of these models are detailed in Table 3. The outcomes, as seen in Figure 13, Figure 14 and Figure 16, indicate that deep learning based segmentation can reduce human errors, especially when comparing the results of models U-net 1, U-net 2, and U-net 3 with images segmented manually by a human. Additionally, images sized 256x256 pixels cover a larger area compared to those of 128x128 pixels. This is evident as the predictions in Figure 15 appear superior to those in Figure 13 and Figure 14. This could explain why the U-net 3 model's predictions of σ phase percentages (17,03%) in Table 3 align more closely with the σ phase percentages (15.14%) in the manually segmented image. In short, deep convolutional neural networks can recognize complex details in images, leading to higher quality segmentation predictions than earlier methods.

Semantic Segmantation using Segmentation-models library

The purpose of using a segmentation models library with the U-Net architecture is to simplify the process of building, training, and deploying U-Net based models for image segmentation tasks. Such libraries often provide pre-defined architectures, pre-trained weights, and utility functions, allowing users to quickly set up and customize U-Net models without starting from scratch. This leads to faster experimentation, improved reproducibility, and potentially better segmentation results. Three different backbones, such as VGG16, InceptionResNetV2, and EfficientNetB7, were used for feature extraction and fine-tuning steps. For these models, 768 original images and their corresponding masks were used, with training conducted over 50 epochs and a learning rate of 0.001. The predictions for the test image, as seen in Figure 17 and Figure 18, indicate that models retrained with pre-existing weights can detect σ phases that human might miss to detect it. Models built on foundational structures, or "backbones", performed better than U-Net models. Using a pre-built U-Net architecture saved time, and re-training pre-existing weights led to better optimization. The best-performing model utilized the EfficientNetB7 backbone.

Its results, displayed in Figure 19, reveal a high-quality identification of σ phase particles, surpassing previous models. This top-performing model was also tested on a different image with a unique structure. The predicted σ phase particles in this image closely matched those in the original, as seen in Figure 21.

5 Conclusions

To summarize, this work demonstrates the feasibility of an effective σ phase microstructural classification in CrMnFeCoNi HEAs using a sequential approach from classical machine learning, where each pixel is classified as σ phase or background with a classifier following feature extraction steps manually, and deep learning methods using U-net convolutional neural network and transfer learning, which eliminates the need for separate segmentation and feature extraction. Data augmentation is required to crop large images into tiny-sized patches in order to increase the quantity of data used to train the models, boosting the accuracy of the models by learning image attributes from large datasets. When deep learning is employed instead of classical machine learning, there is a significant improvement in segmentation accuracy. Aside from the excellent accuracy, we can also accomplish a very rapid prediction. Manually segmenting σ phases using the Matlab image segmenter program takes about 2.5 hours, depending on how complicated the microstructure is, whereas U-net coupled with the transfer learning technique takes about 2 minutes. We pointed out that deep learning segmentation may also reduce human mistakes when the σ phase is not identified in the manual segmenting technique. Additionally, learned deep learning models can be used for various types of material with additional training if the number of training images includes the material's BSE images and associated mask images. Not to mention that not only σ phase particles but also grain boundaries, twin boundaries, carbides, and other microstructure elements may be segmented using the same technique thanks to the user-friendly features of the Segmentation Models library, which allows transfer learning to be used with pre-trained weights.

6 Acknowledgements

I would like to extend my deepest gratitude to Dr.-Ing. Aditya Srinivasan Tirunilai for their invaluable guidance, patience, and unwavering support throughout this research journey. Their insights and expertise have been instrumental in shaping this work. I am also immensely thankful to Prof. Dr. Guillaume Laplanche for their critical feedback and continuous encouragement, which greatly enriched the depth and quality of this study. Additionally, I would like to acknowledge the resources and tools accessed through Dr. Sreenivas Bhattiprolu's GitHub repository "python for microscopists", which played a crucial role in the implementation and validation phases of this project.

References

[1] Jessica Gola, Dominik Britz, Thorsten Staudt, Marc Winter, Andreas Simon Schneider, Marc Ludovici, and Frank Mücklich. Advanced microstructure classification by data mining methods. Computational Materials Science, 148:324–335, June 2018.

[2] George F Vander Voort. Metallography, principles and practice. ASM international, 1999.

[3] Alexandra Velichko. Quantitative 3d characterization of graphite morphologies in cast iron using fib microstructure tomography. 2008.

[4] J Pauly, D Britz, and F Mücklich. Advanced microstructure classification using data mining methods. In TMP-5th International Conference on ThermoMechanical Processing, 2016.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 2012.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015.

[8] Elizabeth A. Holm, Ryan Cohn, Nan Gao, Andrew R. Kitahara, Thomas P. Matson, Bo Lei, and Srujana Rao Yarasi. Overview: Computer Vision and Machine Learning for Microstructural Characterization and Analysis. Metallurgical and Materials Transactions A, 51(12):5985–5999, December 2020.

[9] G. Laplanche, S. Berglund, C. Reinhart, A. Kostka, F. Fox, and E. P. George. Phase stability and kinetics of σ-phase precipitation in CrMnFeCoNi high-entropy alloys. Acta Materialia, 161:338–351, December 2018.

[10] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. PeerJ, 2:e453, 2014.

[11] Gary Bradski. The opencv library. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 25(11):120–123, 2000.

[12] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. Nature methods, 17(3):261–272, 2020.

[13] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In 2008 IEEE conference on computer vision and pattern recognition, pages 1–8. IEEE, 2008.

[14] Byeongkeun Kang and Truong Q. Nguyen. Random Forest With Learned Representations for Semantic Segmentation. IEEE Transactions on Image Processing, 28(7):3542–3555, July 2019.

[15] Ayrildi Zeki. Classical machine learning based image segmentation. https://github.com/Zeki58/Classical-Machine-Learning-based-Image-segmentation/blob/main/0900C_0100h_area2.ipynb, 2023. GitHub repository.

[16] BR Nanditha, A Geetha, HS Chandrashekar, MS Dinesh, and S Murali. An ensemble deep neural network approach for oral cancer screening. 2021.

[17] Ayrildi Zeki. Classical machine learning based image segmentation. https://github.com/Zeki58/Classical-Machine-Learning-based-Image-segmentation/blob/main/transfer_learning_RandomForest.ipynb, 2023. GitHub repository.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015.

[19] Ayrildi Zeki. Classical machine learning based image segmentation. https://github.com/Zeki58/Classical-Machine-Learning-based-Image-segmentation/blob/main/Unet_256.ipynb, 2023. GitHub repository.

[20] Pavel Iakubovskii. Segmentation models. https://github.com/qubvel/segmentation_models, 2019.

[21] Ayrildi Zeki. Classical machine learning based image segmentation. https://github.com/Zeki58/Classical-Machine-Learning-based-Image-segmentation/blob/main/256_Unet_SegmentationModels.ipynb, 2023. GitHub repository.