UNIVERSITÄT
DES
SAARLANDES

Diseño y Entrenamiento de Clasificadores de Bainita y Martensita utilizando Machine Learning

Universidad Nacional de Mar del Plata
Facultad de Ingeniería
Departamento de Ingeniería en Materiales

Autor: Julián Vega
Contacto: julianvega98@gmail.com

Director: Juan Ignacio Moran
Contacto: jmoranunmdp@gmail.com

Co-Director: Björn Ivo Bachmann
Contacto: bjoernivo.bachmann@uni-saarland.de

Proyecto final para optar al grado de Ingeniería en Materiales

Mar del Plata, Agosto 2023

UNIVERSITÄT
DES
SAARLANDES

Diseño y Entrenamiento de Clasificadores de Bainita y Martensita utilizando Machine Learning

Universidad Nacional de Mar del Plata
Facultad de Ingeniería
Departamento de Ingeniería en Materiales

Autor: Julián Vega
Contacto: julianvega98@gmail.com

Director: Juan Ignacio Moran
Contacto: jmoranunmdp@gmail.com

Co-Director: Björn Ivo Bachmann
Contacto: bjoernivo.bachmann@uni-saarland.de

Proyecto final para optar al grado de Ingeniería en Materiales

Mar del Plata, Agosto 2023

Acknowledgments

I would like to extend my sincere gratitude to the following individuals and organizations who have contributed to the completion of my thesis:

- My family and my girlfriend, for their unconditional support, encouragement and understanding throughout this journey.
- UNMdP (Universidad Nacional de Mar del Plata) and my profesors, for providing me with five years of free and high-quality education.
- EUSMAT and UNMdP, for funding and organizing my scholarship, I.DEAR, which allowed me to conduct my thesis research in Germany.
- My supervisor, Björn Bachmann, who invited me to collaborate with him, offered me the opportunity to work on my thesis topic and provided guidance throughout the entire process.
- My director, Juan Ignacio Morán, who provided guidance throughout the process and played a crucial role in helping me prepare and present my work at UNMdP.
- Aktien-Gesellschaft der Dillinger Hüttenwerke, for providing me with the necessary samples and equipment for my research.
- MECS (Material Engineering Center Saar), for granting me access to their laboratories.
- Flavio Soldera and Silvia Simison, for their coordination and assistance during my scholarship program.
- David Cagni, for accompanying me during my time in Germany and assisting in obtaining the data.

- My friends and colleagues at UNMdP, for their camaraderie, shared experiences and support.

I am deeply grateful for the support, guidance, and contributions of these individuals and organizations, as they have played a pivotal role in the successful completion of my thesis.

Contents

Resumen

El acero ha desempeñado un papel fundamental en el avance de la sociedad moderna. Su importancia es evidente en el aumento exponencial de la demanda y producción presenciado en las últimas décadas. La excelente combinación de propiedades, como alta resistencia, ductilidad, tenacidad y resistencia a la corrosión, junto con un precio relativamente bajo, hace que el acero sea adecuado para una amplia gama de aplicaciones en varios sectores.

La demanda de aceros de alta calidad ha llevado a científicos e ingenieros a desarrollar nuevas rutas de procesamiento con el fin de mejorar sus propiedades. Estas nuevas rutas de procesamiento tienden a formar microestructuras más finas y complejas.

Los aceros de alta calidad suelen estar compuestos por una mezcla compleja de varios microconstituyentes, incluyendo ferrita, bainita, martensita y eventualmente austenita retenida [1, 2]. La presencia de múltiples fases de diferentes tamaños permite la producción de aceros con propiedades específicas para cada aplicación. Para garantizar el control de calidad y facilitar el desarrollo de materiales con propiedades deseadas, es crucial caracterizar objetivamente estas fases.

El análisis y la caracterización microestructural representan una parte esencial del desarrollo de materiales y el control de calidad. Debido a la creciente complejidad de las microestructuras, los métodos tradicionales de análisis y caracterización a menudo no producen resultados satisfactorios. Los enfoques basados en imágenes de microscopio óptico (LOM, por sus siglas en inglés) son cada vez menos eficaces, especialmente para el análisis cuantitativo de materiales con microestructuras más finas y un mayor número de microconstituyentes [1, 3, 4]. Además, la caracterización de estos aceros complejos por parte de expertos

es altamente subjetiva y puede depender de su experiencia y expectativas[4, 5, 6].

La microscopía correlativa es una técnica poderosa que permite la caracterización objetiva y reproducible de muestras mediante el uso de diferentes métodos de contraste en una ubicación específica de la muestra [3, 5]. El proceso de alinear imágenes en el mismo sistema de coordenadas se denomina registro de imágenes. Al combinar información de diferentes fuentes físicas, la microscopía correlativa proporciona una comprensión más profunda de la microestructura de la muestra.

Un enfoque interdisciplinario que utiliza *machine learning* (ML) proporciona una forma adicional de reducir el grado de subjetividad en la caracterización microestructural [5]. Los algoritmos de ML pueden analizar grandes conjuntos de datos para identificar patrones y relaciones que pueden no ser fácilmente apreciables para un observador humano, mejorando así la objetividad y reproducibilidad del análisis. La aplicación de herramientas modernas de informática, incluido el ML, en la ciencia de materiales es una tendencia creciente. La combinación de microscopía correlativa y ML representa un área prometedora de investigación para avanzar en la comprensión y el desarrollo de materiales.

El objetivo principal de esta tesis es diseñar y entrenar modelos de *machine learning* para la clasificación objetiva de las fases de acero bainita y martensita. Para ello se utilizó un enfoque correlativo basado en imágenes de microscopio óptico (LOM), microscopio electrónico de barrido (SEM) y mapas de Electron Backscatter Diffraction (EBSD).

Se utilizaron dos grupos de probetas con composiciones distintas. Las composiciones se detallan en la tabla 3.1. Cada grupo de probetas consistió en cinco muestras enfriadas a diferentes velocidades, lo que resultó en microestructuras diferentes. Las velocidades de enfriamiento de cada probeta se encuentran en las tablas 3.2 y 3.3. Estas probetas se montaron en un material apto para el análisis en SEM, se desbastaron y pulieron hasta 1 µm. Los pasos de preparación metalográfica se explican en la tabla 3.4. Luego, se seleccionó cuidadosamente una Región de Interés (ROI) para cada muestra y se delimitó mediante indentaciones de una máquina de ensayos de dureza. Finalmente, se adquirieron imágenes utilizando LOM y SEM. Además, se obtuvieron mapas EBSD para cada muestra, lo que proporcionó información valiosa sobre la

orientación cristalográfica.

Luego de obtener las imágenes, se registraron utilizando el programa de código abierto Fiji [7]. Para ello, se seleccionaron manualmente puntos distintivos en cada micrografía y se correlacionaron entre sí. Una vez seleccionados estos puntos, se realizó el registro utilizando el complemento bUnwarpJ [8]. En primer lugar, se registraron las imágenes LOM al mapa de *Image Quality* (IQ) de EBSD, lo cual permitió también superponer el resto de los mapas de EBSD, como el *Confidence Index* (CI), el *Kernel Average Misorientation 1* (KAM1), el *Kernel Average Misorientation 3* (KAM3), el *Grain Orientation Spread* (GOS) y el *Grain Average Misorientation* (GAM). Posteriormente, se registraron las imágenes SEM a las LOM, logrando así una superposición de todas las imágenes. A partir de esta superposición, se creó manualmente una máscara de bordes de grano de austenita previa [9], que permitió extraer objetos individuales de bainita y martensita de cada micrografía.

Se desarrollaron scripts e interfaces gráficas de usuario (GUI) en Python para la adquisición y manipulación eficiente de conjuntos de datos. Estas GUI permiten la extracción rápida de objetos de los diferentes métodos. También se incorporaron características adicionales para que el software fuera adecuado tanto para la creación de conjuntos de datos como para el entrenamiento y la aplicación de modelos de forma local.

Se entrenaron múltiples modelos de *machine learning* para ambos conjuntos de muestras. Los objetos utilizados para entrenar los modelos son matrices de una cierta altura y anchura, que contienen los seis parámetros de EBSD: IQ, CI, KAM1, KAM3, GOS y GAM. El proceso de selección del modelo y la optimización de hiperparámetros involucró una comprensión integral de diversas técnicas de *machine learning*.

Inicialmente, se entrenaron modelos de Random Forest (RF) y Support Vector Machine (SVM) debido a su facilidad de implementación utilizando los módulos de Python disponibles. Si bien estos métodos no están específicamente diseñados para esta tarea, proporcionaron una opción práctica para una implementación rápida y eficiente. La precisión de cada modelo se encuentra en la tabla 4.1. Los modelos mostraron una precisión satisfactoria, aunque no excepcionalmente alta. Cabe destacar que su rendimiento puede disminuir para tareas más complejas, como la clasificación de múltiples clases.

Posteriormente, se realizó una búsqueda para identificar una arquitec-

tura de red neuronal convolucional (CNN) que pudiera resolver eficazmente la tarea en cuestión. A través de experimentos y ajustes de varias arquitecturas e hiperparámetros, se determinó una configuración que mostró resultados prometedores. Esta arquitectura se utilizó para ambos conjuntos de muestras, variando solo la forma de entrada de los objetos. La precisión de cada modelo se encuentra en la tabla 4.1. Los modelos lograron una precisión y reproducibilidad notablemente altas. Aunque estos modelos ya se pueden utilizar para la clasificación objetiva y reproducible de bainita y martensita, cabe mencionar que se pueden realizar mejoras adicionales, como incorporar un conjunto de datos más amplio o realizar una selección de características que podría mejorar el rendimiento.

Además, esta arquitectura demuestra un potencial prometedor para abordar tareas más complejas de mayor relevancia en aplicaciones científicas e industriales. En particular, en la clasificación de múltiples fases de acero, incluyendo diferentes tipos de ferrita, bainita, martensita y perlita, lo cual se explorará en investigaciones futuras.

En resumen, este proyecto logró caracterizar de manera efectiva y objetiva las microestructuras de dos grupos de muestras con distintas composiciones. Se utilizó un enfoque correlativo que combinó imágenes de LOM y SEM con mapas de EBSD, lo que permitió una comprensión más profunda de la microestructura. Se desarrollaron scripts y GUIs en Python para la adquisición y manipulación eficiente de los datos. Se entrenaron modelos de *machine learning*, incluyendo RF, SVM y CNN, y se logró una alta precisión en la clasificación de bainita y martensita. Este trabajo sienta las bases para futuras investigaciones en la clasificación de múltiples fases de acero y tiene aplicaciones potenciales en diversos campos científicos e industriales.

Abstract

This thesis focuses on the design and training of machine learning models to successfully separate bainitic and martensitic objects in steel microstructures. The study involves the investigation and characterization of two groups of steel samples with distinct compositions, each comprising five samples subjected to varying cooling rates.

The objective of this research is to develop an objective and reproducible method for characterizing bainite and martensite microstructures. To achieve this, an interdisciplinary approach combining modern microscopy techniques with machine learning is adopted. Correlative microscopy is utilized to gain a deeper and more objective understanding of the microstructure, as well as to establish ground truths. This approach involves acquiring images from Light Optical Microscopy (LOM), Scanning Electron Microscopy (SEM), and Electron Backscatter Diffraction (EBSD) maps.

Computer vision and machine learning tools are employed to create models for the classification of bainite and martensite objects. Random Forest (RF), Support Vector Machines (SVM), and Convolutional Neural Networks (CNN) models are trained using numerical arrays of the EBSD data associated with each object.

By combining advanced microscopy techniques with artificial intelligence, this thesis aims to provide a reliable approach for the objective and reproducible characterization of bainite and martensite in steel microstructures. The performance and effectiveness of the developed models will be evaluated, providing insights into the potential application of similar techniques in materials science and industrial applications.

Chapter 1

Introduction

Steel has played a critical role in the advancement of modern society. Its significance is evident in the exponential increase in demand and production witnessed in recent decades. The excellent combination of properties such as high strength, ductility, toughness and corrosion resistance, together with a relatively low price, makes steel suitable for a wide range of applications in various sectors.

The demand for high quality steels has been the driving force for scientists and engineers to develop new processing routes. In trying to improve the properties of steel, these new processing routes tend to form finer and more complex microstructures.

High-quality steels are typically composed of a complex mixture of several microconstituents, including ferrite, bainite, martensite, and possibly retained austenite [1, 2]. The presence of multiple phases of different sizes allows for the production of steels with tailored properties to meet specific application requirements. To ensure quality control and facilitate the development of materials with desired properties, it is crucial to objectively characterize these phases.

Microstructural analysis and characterization represents an essential part of material development and quality control. Due to the growing complexity of microstructures, traditional analysis and characterization methods often fail to produce satisfactory results. Approaches that rely on light-optical microscope (LOM) images are becoming less useful, especially for the quantitative analy-

sis of materials with increasingly fine microstructures and a larger number of micro-constituents [1, 3, 4]. Furthermore, the characterization of these complex steels by experts is highly subjective and can depend on their experience and expectations [4, 5, 6].

Correlative microscopy is a powerful technique that allows for objective and reproducible characterization of samples by using different contrasting methods at a specific sample location [3, 5], which are then combined with each other. The process of aligning image-like data to the same coordinate system is referred to as image registration. By combining information from different sources, correlative microscopy provides a deeper understanding of the microstructure of the sample.

An interdisciplinary approach using machine learning provides an additional way to reduce the degree of subjectivity in microstructural characterization [5]. Machine learning algorithms can analyze large sets of data to identify patterns and relationships that may not be readily apparent to a human observer, thus enhancing the objectivity and reproducibility of the analysis. The application of modern computer science tools, including machine learning, in materials science is a growing trend [5]. The combination of correlative microscopy and machine learning represents a promising area of research for advancing the understanding and development of materials.

The primary objective of this thesis is to design and train machine learning models that are capable of accurately classifying bainite and martensite objects in a reproducible and objective manner. To achieve this goal, a correlative approach will be employed, combining data from light optical microscopy (LOM), scanning electron microscopy (SEM), and electron backscatter diffraction (EBSD). A mask of the prior austenite grain boundaries will be created to obtain objects. Two different machine learning models, Support Vector Machine and Random Forest, will be trained for this purpose. In addition, a Convolutional Neural Network will be designed and trained to classify the objects. The performance of these three models will be evaluated and compared to identify the most suitable one for accurate and reliable classification of bainite and martensite objects.

Chapter 2

Theoretical foundations

2.1 Fundamentals of Materials Science

2.1.1 Steel

Alloys that contain iron and carbon are collectively referred to as steel, and they can be combined with various alloying elements that have a significant impact on the final material properties, even in small percentages. However, the carbon content of the alloy is the most significant factor affecting its properties. Alloys with up to 2% carbon are classified as steel, while alloys with carbon content exceeding 2% are classified as cast iron.

The properties of steel can vary widely depending not only on its chemical composition but also on the processing route and thermomechanical treatments it undergoes. These processes take advantage of the solid state transformations that occur in steel to generate different microstructures with different properties [10]. Microstructure refers to the arrangement of atoms and grains within a material and is determined by its composition and processing history. The microstructure of steel stores information about the processing it has undergone and plays a crucial role in determining its mechanical, physical, and chemical properties. Therefore, understanding the microstructure of steel is crucial in designing materials with specific properties for different applications.

As the demand for high-quality steels continues to grow, their microstruc-

ture has become increasingly complex and refined, posing a significant challenge for its characterization, which is crucial for materials development and quality control. Classical methods, such as light optical microscopy (LOM), are approaching their technical limits for characterizing these fine structures [4]. Additionally, the increasing complexity makes it more difficult for experts to classify it objectively, leading to a growing need for objective and automated characterization methods.

2.1.2 Phases of steel

Iron exhibits allotropy, which refers to the ability of a chemical element to have a stable existence in more than one crystal form. The stable state of an element or compound is determined by the lowest molar Gibbs free energy at the pressure and temperature of interest [11]. At room temperature and pressure, the most stable form of iron is α-iron or ferrite, which possesses a body-centered cubic (bcc) crystal structure. However, at higher temperatures, ferrite undergoes a solid-state transition and transforms into γ-iron or austenite, which has a face-centered cubic (fcc) crystal structure. At even higher temperature, austenite transition back to body-centered cubic structure called δ -iron. Furthermore, at very high pressures, α - iron changes into a hexagonal close-packed (hcp) structure, known as ϵ -iron. However, the latter two structures have little significance in technical applications.

There are two major groups of solid-state transformations: reconstructive and displacive. In reconstructive transformations, all bonds are broken, and the atoms are rearranged into a new pattern, which means that they occur with the diffusion of atoms [10]. Displacive transformations, on the other hand, maintain all bonds but deform the structure, for example, through shear, to form a new crystal structure. Displacive transformations are diffusionless [10].

Carbon is a crucial element in the formation and constitution of steel phases. The carbon content and its maximum solubility in the iron lattice significantly influence the final phases of steel. Carbon atoms are smaller than iron atoms, which enables them to enter the α-iron and γ-iron lattices as interstitial solute atoms. However, as the atomic size of carbon is larger than the size of available interstices, it creates distortions in the lattice that result in a strain energy. The resulting strain field impedes the mobility of dislocations, thereby

increasing the steel's strength [10]. This strengthening mechanism is called solid solution strengthening and occurs with various alloying elements.

The solubility of carbon in ferrite and austenite is determined by their crystal structures. Ferrite has a bcc structure with six tetrahedral holes and three octahedral holes, with radii relative to iron of 0.29 r and 0.15 r, respectively [10]. Although the tetrahedral holes are larger, it has been observed that carbon occupies the octahedral holes. This is because in the case of tetrahedral interstices, carbon would displace four nearest-neighbor iron atoms, requiring more strain energy than the two nearest-neighbor displacements produced in the octahedral interstices [10]. The fcc structure of austenite, although more closely packed, has larger holes than the bcc structure, which makes the solubility of carbon in austenite in equilibrium with ferrite greater. This structure has two tetrahedral interstices and one octahedral interstice, with radii relative to iron of 0.23 r and 0.41 r, respectively [10]. If the carbon content exceeds the solubility limit of the solid solution, the carbon is precipitated as an iron carbide (Fe_3C) called cementite. This compound has an orthorhombic crystal structure and a carbon solubility of 6.67%. Due to its crystalline structure and chemical composition, cementite is harder and more brittle compared to the solid solution.

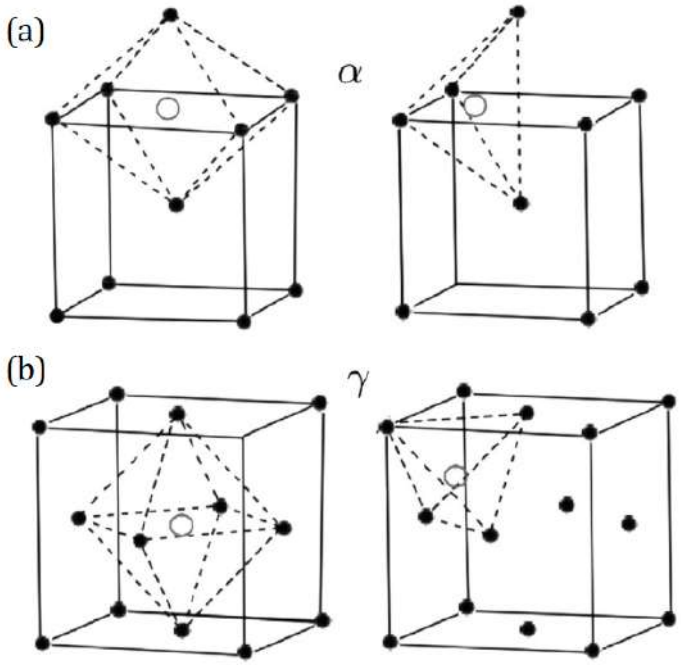

Figure 2.1: (a) Bcc structure of ferrite showing an octahedral interstice on the left and a tetrahedral interstice on the right; (b) Fcc structure of austenite showing an octahedral interstice on the left and a tetrahedral interstice on the right. [10]

The Fe-C equilibrium diagram in Figure 2.2 summarizes the prevailing phases at each temperature and as a function of carbon content. It is important to distinguish between stable and metastable systems. Graphite is the stable phase with the lowest free energy, but its formation requires very high diffusion times. In most technical application, cementite forms instead, which is a metastable phase.

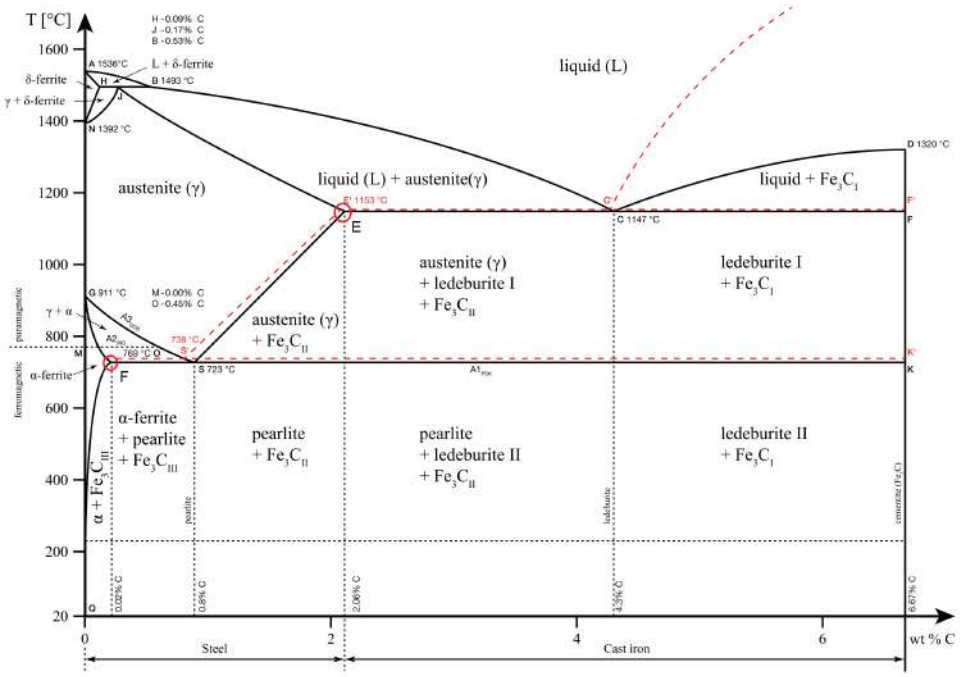

Figure 2.2: Iron - Carbon diagram. [12]

The difference in carbon solubility between austenite and ferrite is reflected in the much larger phase field of austenite. At 1147 °C, when in equilibrium with liquid and cementite (point E in Figure 2.2), the maximum solubility of carbon in austenite is about 2 wt%. On the other hand, the α-iron phase field is severely limited, with a maximum carbon solubility of only 0.02 wt% at 727 °C when ferrite is in equilibrium with austenite or cementite (point P in Figure 2.2) [10]. This significant difference in solubility is crucial for heat treatments that aim to produce supersaturated solid solutions by rapidly cooling from the austenitic field.

Figure 2.2 shows the eutectoid point at which the Fe-C system undergoes a solid-state transformation from austenite to ferrite and cementite. During the eutectoid transition, one solid phase converts into two solid phases, typically exhibiting a lamellar structure. In the Fe-C system, the eutectoid point is located at 727 °C and 0.76 wt% carbon, which is the lowest temperature at which austenite is stable. The transformation occurs via a reconstructive solid-state reaction that involves the successive nucleation and growth of ferrite and

cementite lamellae. Nucleation occurs at austenite grain boundaries or other high-energy zones with a local depletion of carbon, resulting in the formation of low-carbon ferrite. This leaves an adjacent zone with a high carbon content where cementite nucleation and growth are favored. The resulting structure is called a pearlite colony, which consists of an interpenetrating bicrystal of ferrite and cementite. In planar sections, the phases appear as a lamellar structure [10]. The thickness of each lamella depends on the cooling rate, as the transformation relies on the diffusion of iron and carbon atoms. Higher cooling rates hinder diffusion and therefore, thinner lamellae are formed.

The microstructure at the eutectoid point exhibits a purely pearlitic character. At lower carbon concentration, the austenite transforms into ferrite, with the austenite becoming enriched in carbon until it reaches the eutectoid concentration, at which point pearlite is formed. This microstructure is called hypoeutectoid and is characterized by the presence of ferrite grains and pearlite colonies. In contrast, when the carbon concentration exceeds 0.76 wt%, cementite forms first, reducing the carbon content in the austenite until it reaches the eutectoid composition and transforms into pearlite. This results in a microstructure known as hypereutectoid, which contains cementite grains and pearlite colonies. The samples used in this work have low and medium carbon content of around 0.088 wt% and 0.219 wt%, respectively, and thus exhibit a hypoeutectoid composition.

The microstructures discussed so far arise under moderately low cooling rates that favor reconstructive transformations. However, the final phases of steels are strongly dependent on the cooling rates. The actual microstructures obtained are metastable and often differ from those predicted by the equilibrium phase diagram. When the cooling rate is high, only displacive transformations are possible, and many different phases can be formed depending on this parameter.

Steel can exhibit various phases, including different types of ferrite, bainite, pearlite, and martensite, depending on its composition and cooling rate. However, it is crucial to acknowledge the existence of multiple classification schemes, which frequently contradict one another and lack clarity. Taking these challenges into consideration, this work will simplify the analysis by placing a primary emphasis on martensite and bainite, as the main focus is on the methodical classification based on EBSD data.

The time-temperature-transformation (TTT) diagram illustrates the final phases that form based on the cooling rate. A general example of a TTT diagram can be seen in Figure 2.3. The formation mechanisms and characteristics of bainite and martensite will be explained in detail in the following sections as they are relevant to the current work.

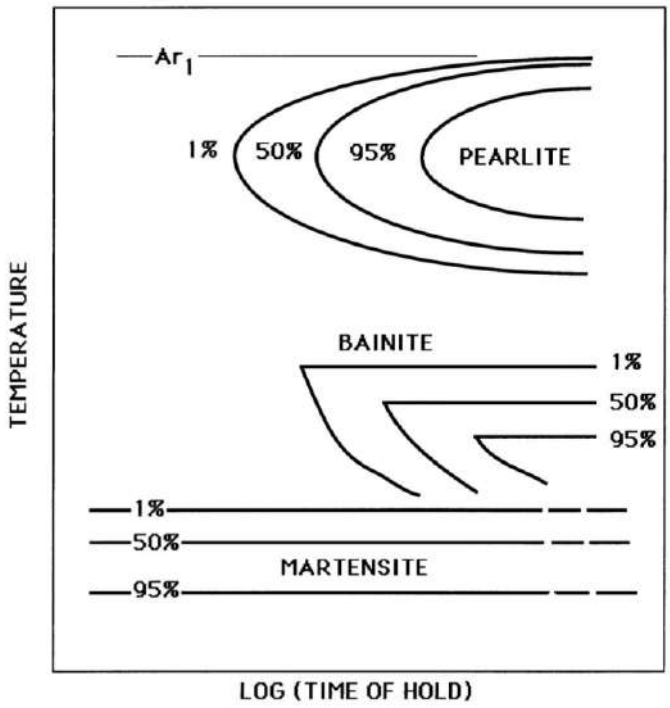

Figure 2.3: Schematic illustration of a TTT diagram [13].

2.1.3 Martensite

Martensite is a hard and brittle phase that forms in steel when austenite is rapidly quenched to room temperature. This process involves such high cooling rates that the diffusion of atoms becomes negligible. Unlike ferrite or pearlite, which form through diffusional mechanisms, martensite forms through a deformation of the austenite lattice without any diffusion of atoms, i.e., through a displacive mechanism. The fcc structure of austenite is transformed into a super carbon-saturated bcc structure. The plates of martensite can grow at

speeds that approach the speed of sound in the material, reaching up to 1100 m/s, whereas the maximum velocity measured for a reconstructive transformation is only 80 m/s [10]. Furthermore, the final composition of martensite is the same as that of the parent austenite, which further demonstrates that this transformation is indeed diffusionless [10].

The interface between austenite and martensite is known as the habit plane. When the transformation is unconstrained, this interface plane is typically flat. However, when the transformation is constrained by its surroundings, such as in a polycrystalline material, strain energy minimization can introduce some curvature to the interface [10]. To achieve rapid growth at low temperatures, this interface must be glissile, meaning it can move with the glide of dislocations [14].

Since the formation of martensite involves the coordinated movement of atoms, the lattice orientation of martensite and austenite are closely related. This coordinated motion of atoms also implies that the transformation cannot be sustained across austenite grain boundaries [15].

The rearrangement of atoms during the formation of martensite leads to a macroscopic change in the crystal's shape, resulting in a large shear component (≈ 0.22) and a small dilatational strain (≈ 0.03) normal to the habit plane [10, 14]. This strain required to transform the fcc structure to the bcc structure is known as the Bain strain. It involves a compression along the z-axis of the austenite crystal and uniform expansion along the x and y axes.

The Bain strain predicts a rational orientation relationship between the parent and product lattices, which contradicts the observed irrational orientation relationships. Additionally, it does not leave any line invariant, which violates the essential condition for a glissile interface. However, the combination of the Bain strain and a rigid body rotation satisfies these conditions, enabling accurate prediction of the irrational orientation relationship. These orientation relationships can be utilized to reconstruct the parent grains [16]. The combination of these two transformations, however, only yields an invariant-line strain and not the observed invariant-plane strain that occurs during the transformation of austenite to martensite. The phenomenological theory of martensite crystallography solves this problem by adding a second homogeneous shear to the Bain strain. However, this second shear leads to the wrong shape deformation. To correct this, an inhomogeneous lattice-invariant deformation, such as

slip or twinning, must cancel out the macroscopic shape-changing effect of the second shear, as illustrated in Figure 2.4 [10, 14].

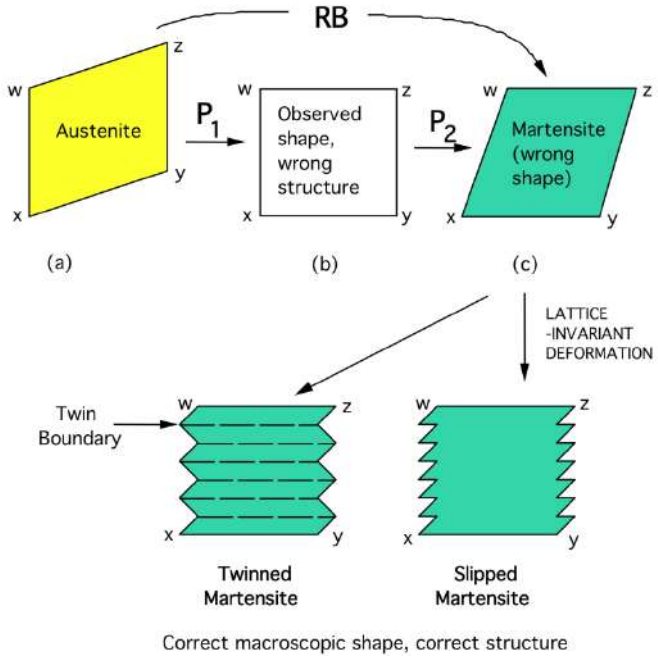

Figure 2.4: Summary of the phenomenological theory of martensite crystallography [14].

Figure 2.5 depicts the free energy of ferrite and austenite as a function of carbon concentration at a given temperature T_1. The equilibrium composition at this temperature is obtained by drawing a tangent line to the minimum of both curves, representing a transformation where carbon diffusion is possible. At a certain point, ferrite and austenite have the same carbon composition and free energy. Collecting this information for different temperatures gives the T_0 curve, indicating the threshold where the martensitic transformation is thermodynamically possible or not. For lower carbon concentrations, a transformation from austenite to ferrite without altering composition leads to a decrease in free energy, while for higher concentrations, this transformation causes an increase in free energy, making it thermodynamically infeasible.

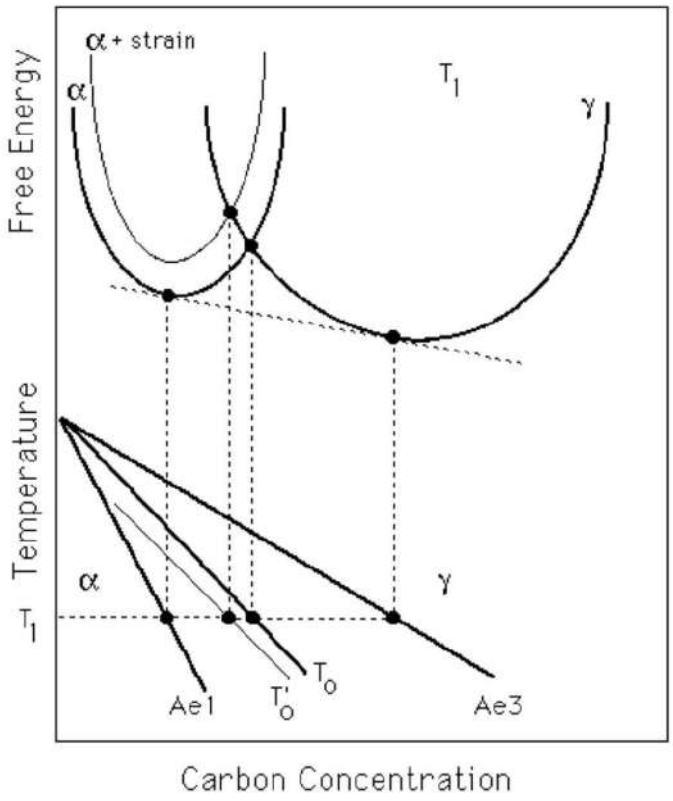

Figure 2.5: Schematic illustration of the origin of the T_0 curve on the phase diagram.[17].

The α+ strain curve takes into account the energy associated with the deformation energy resulting from the transformation. By including additional energy terms, such as those associated with twin interfaces or α/γ interfaces, the Ms temperature, which marks the onset of the transformation, can be determined.

The martensitic transformation is considered an athermal process, meaning that the volume fraction of martensite does not increase with time when held at a temperature below Ms. Instead, the transformation rate is primarily determined by the degree of undercooling below the martensite start temperature [10]. This is because nuclei that are easier to form are triggered at small undercooling, while progressively more difficult-to-form nuclei are triggered as the

undercooling increases [10]. There is no martensite-finish temperature, since the martensite volume fraction tends to 1 when the temperature tends to 0, but for convenience, Mf is defined at the point where 95% of the transformation is completed. This also implies that there is always a certain amount of austenite that remains untransformed at room temperature, which is called retained austenite [10].

The carbon concentration is a critical factor influencing the characteristics and properties of martensite. Firstly, the morphology and crystallography of martensite are highly sensitive to carbon content. At low carbon concentrations, martensite exhibits a lath or plate-like morphology with low-misorientation boundaries between each lath. However, at higher carbon concentrations, the plates of martensite become more lenticular, and the habit planes change at around 0.5 wt% [10]. Secondly, the tetragonality of the bcc lattice increases with carbon content, leading to a transformation to a body-centered tetragonal lattice (bct) at high concentrations. This distortion produces an asymmetrical strain field with a high shear component that strongly interacts with dislocations, hindering their mobility and increasing hardness. Thirdly, Ms decreases with carbon concentration, implying that the amount of retained austenite also increases with carbon content. Finally, the carbon concentration is perhaps the most crucial strengthening factor in martensite because it increases the tetragonality of the lattice. Additionally, the subdivision of austenite into either laths or lenticles creates low-angle sub-boundaries that impede the mobility of dislocations, thereby contributing to an increase in hardness [10].

2.1.4 Bainite

The definition and formation mechanism of bainite have been subjects of controversy since its inception [18]. The general microstructural definition describes bainite as a nonlamellar eutectoid decomposition product [19, 20]. However, three conflicting definitions exist, making it impossible for all of them to be simultaneously true. The two primary theories regarding the formation mechanism contradict each other, as one proposes a diffusion-controlled transformation while the other favors a displacive, diffusionless transformation [18].

In recent years, the theory of diffusionless, displacive transformation has gained increasing scientific evidence and demonstrated superior predictive

power. In fact, only this theory has been applied to alloy design [18]. Therefore, this theory will be adopted and explained in this work as the preferred approach.

Bainite and martensite are both phases with a bcc lattice that result from the decomposition of austenite through a displacive transformation. Both phases cause an invariant-plane strain deformation. In bainite, the shear component is approximately 0.26 and the dilatation strain normal to the habit plane is approximately 0.03 [13]. As in martensite, there is a synchronized movement of atoms that confines the transformation within the austenite grain. However, bainite forms at a temperature that is above the Ms temperature but below that at which fine pearlite forms, allowing for some mobility of carbon atoms.

Bainite formation begins with the heterogeneous nucleation of carbon supersaturated ferrite in a plate-like shape at austenite grain boundaries. This nucleation process is diffusionless. However, the transformation temperature for bainite is higher than that of martensite, allowing carbon atoms to partition into the residual austenite, leading to the subsequent nucleation of carbides. Depending on the temperature, these carbides can precipitate in two forms, giving rise to two types of bainite.

At higher temperatures, typically between 400°C and 500°C, carbon mobility is high enough for it to precipitate in the adjacent austenite, forming Upper Bainite (UB). Conversely, when the transformation occurs between 250°C and 400°C, there is not enough time for all the carbon to diffuse to the austenite. As a result, some carbides precipitate within the plates in the form of finely divided cementite. However, there is also some precipitation of cementite from carbon-enriched austenite outside the plates. This type of bainite is known as Lower Bainite (LB). Figure 2.6 represents the precipitation of these carbides in both types of bainite

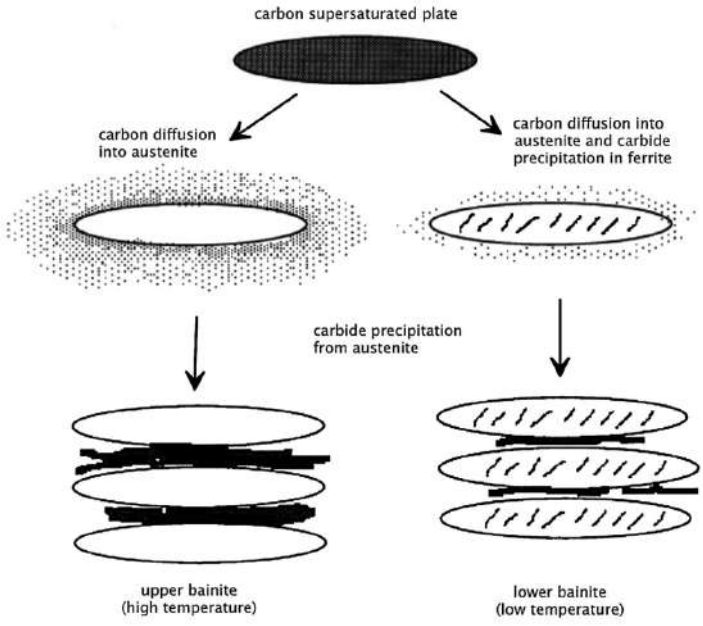

Figure 2.6: Schematic representation of the precipitation of carbides in Upper and Lower bainite [17].

The precipitation of coarse cementite particles between plates in UB can lead to a reduction in toughness compared to LB. Moreover, LB exhibits a slightly finer microstructure and thinner inter-plate carbides compared to UB, which results in higher strength and toughness at the same time.

Both upper and lower bainite consist of clusters of ferrite plates separated by cementite, and occasionally by untransformed austenite or martensite. These clusters, called sheaves, are made up of multiple sub-units that are interconnected in three dimensions and share a common crystallographic orientation. The sub-units within a sheaf have uniform dimensions because they grow until they reach a limiting size. The majority of new sub-units form near the tips of existing sub-units rather than along their sides, as shown in Figure 2.7 [13].

The sub-units in upper bainite are typically about 10 µm in length and 0.2 µm in thickness, while those in lower bainite are slightly smaller. Due to the limited resolution of optical microscopes, individual bainite plates cannot be resolved as their size is smaller than the wavelength of visible light, which

is around 0.5 µm [13, 21]. Nonetheless, an ensemble of these plates can be observed.

It is this microstructure that gives bainite its high strength due to its ultrafine grain size. The width of the plates determines the mean free slip distance of the dislocations, making the effective grain size less than one micrometer. This is significant because grain refinement is the only method that can increase both hardness and toughness simultaneously. This phenomenon, in which strength increases as grain size decreases, is known as the Hall-Petch effect and is widely used to increase the strength of steel.

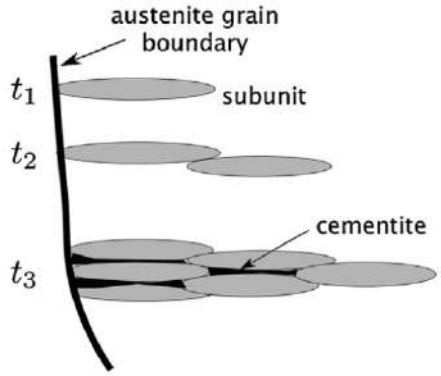

Figure 2.7: Schematic illustration of the evolution of the structure of sheaves in bainite over time [10].

Since bainite forms at higher temperatures than martensite, the parent austenite is weaker and unable to elastically accommodate large shape deformations. Instead, it undergoes plastic deformation in the region adjacent to the bainite [10]. During the early stages of growth, the bainite is able to absorb some of the dislocations created by this plastic relaxation, resulting in a certain density of dislocations within the bainite but also ahead of the interface. However, the high density of dislocations at the interface eventually prevents it from moving because, in a displacive transformation, the interface must be glissile, which means that the dislocations must be able to slide freely. As a result, the bainite plate grows until it is stopped by the plastic deformation of the austenite, resulting in a size smaller than the austenite grain size. Further transformation takes place through the formation of new plates, leading to the

development of the characteristic sheaf morphology.

It has been shown that there is a well-defined temperature B_s above which no bainite will form. The volume fraction of bainite increases with undercooling, similar to the formation of martensite. During isothermal transformation, the fraction of bainite formed follows a sigmoidal function of time, eventually reaching an asymptotic limit that remains constant even when significant amounts of austenite are still present after prolonged heat treatment [10]. This is because the partitioning of carbon increases the carbon content in austenite until it reaches the composition of the T_0 curve, as explained in Figure 2.5. At this point, the displacive mechanism is no longer thermodynamically possible, and further transformation can only occur by increasing undercooling.

2.1.5 Microstructural Analysis

Etching

To characterize the microstructure, it is essential to distinguish the microstructural constituents through contrasting techniques. Microstructural contrasting refers to any technique that enables the identification and differentiation of the microstructural constituents. Traditional characterization methods like LOM require some topography on the sample's surface to be able to distinguish the microstructural constituents.

Electrochemical etching is a widely used method for structural etching, which generates contrast in metallographic specimens. During this process, reduction and oxidation reactions occur between the etchant and the sample's surface [22]. The different phases in the sample are selectively attacked based on their electrochemical potential [23], which is determined by the concentration of alloying elements, primarily carbon. The result is a topographical contrast that can be observed under an optical microscope.

The electrochemical potentials of the constituents depend not only on composition but also on physical inhomogeneities, such as inclusions, grain boundaries, or crystal orientations. These areas usually have higher potentials and are more severely attacked, revealing the grain boundaries.

Precipitation or color etching is another type of etching technique that operates on a different physical principle. This method involves the reaction

between the etching solution and the metal surface, leading to the formation of stable precipitates with different thicknesses on the polished surface. The thickness of these layers is determined by the chemical potential and can vary from 40 to 500 nm [24]. When light reflects off the sample, the presence of these precipitate layers causes destructive interference, resulting in a diverse range of colors visible when observed through an optical microscope.

Nital is the most common electrochemical etchant used to reveal the microstructure of carbon steels. It consists of a solution of concentrated nitric acid (HNO_3) in ethyl alcohol. Most recipes in the literature implement a solution of 2% nitric acid, but the concentration can range between 1% and 5% [23]. In the context of this work, we used a mixing ratio of 100 ml ethanol with 2.5 ml 65% nitric acid.

Classical Microstructural Analysis

The classical microstructural analysis of metals and alloys involves several steps, including metallographic preparation, contrasting, and examination under an optical microscope. The conventional approach for evaluating microscopic images relies heavily on visual assessment performed by an expert. However, this method is associated with a significant degree of subjectivity that depends on the metallographer's experience and training, as well as the choice and quality of the contrasting method.

With the advent of modern materials with increasingly finer microstructures, traditional methods have become less effective for accurate analysis. The conventional LOM has reached its resolution limit, making it difficult to resolve, for example, individual lamellae of pearlite or the thin sub-units of bainite. As a result, the structure of pearlite appears as dark colonies, and important substructure information is lost due to destructive interference.

To overcome the low resolution of LOM, SEM has emerged as a complementary imaging technique. SEM can achieve a resolution of up to 1 nm, making it a high-resolution microscopy technique, unlike the LOM, which has a resolution limit of approximately 350 nm dictated by the Abbe diffraction limit. SEM images can provide greater detail, making it essential for unambiguous microstructural characterization of modern steels. However, SEM imaging comes with the challenges of being costly, time-consuming, and more complex in terms

of the contrast mechanism, thereby making the acquisition of proper images more difficult. Nevertheless, due to the advantages of its higher resolution, SEM imaging remains an essential component in this context.

Modern Approaches to Microstructural Analysis

Electron backscatter diffraction (EBSD) is a widely used measurement technique in materials science that has gained widespread popularity in recent decades. It is carried out in a SEM equipped with special EBSD detectors. During EBSD measurement, an electron beam is incident on the sample, and the electrons interact with the atomic nuclei, causing both elastic and inelastic interactions. Some of the elastically scattered electrons are deflected back towards the sample surface and interact with the crystal lattice of the sample, following Bragg's law, and create a diffraction pattern known as a Kikuchi pattern. This pattern can be compared with a database to obtain information about the crystal structure and orientation [25, 26].

By scanning EBSD over a selected sample area, a range of crystallographic information can be obtained, from which EBSD maps can be constructed. These maps serve as an additional source of information. Grain boundaries can be defined by setting a threshold value for the misorientation angle. The crystallographic orientation is represented in the Inverse Pole Figure (IPF) using a triangle color scheme. The quality of the diffraction pattern, which indicates lattice imperfections, is another important measure provided by EBSD. Despite the additional information provided by EBSD, differentiating between ferrite and bainite proves challenging due to their identical crystal structure. However, the utilization of approaches that analyze the image quality of the diffraction pattern has demonstrated the ability to differentiate between these two phases [1, 21]. This is primarily because bainite exhibits higher dislocation densities compared to ferrite, attributed to its distinct formation mechanism.

There are various metrics that provide information about the quality of diffraction patterns. One common approach is to use the Image Quality (IQ) metric, which describes the quality of a diffraction pattern. An IQ map can be constructed by mapping the IQ values measured for each diffraction pattern obtained during scanning to a gray or color scale [27]. This approach has been extensively used in different studies, and several works have reported the usefulness of the IQ map in analyzing the microstructure of materials [1, 28, 27, 29,

30]. However, it is important to note that IQ values can be affected by various factors, including sample preparation and contamination, surface topology, and beam conditions [2, 31]. These factors can lead to variability in results and make it challenging to achieve reproducible and objective characterization.

Additionally, the Confidence Index (CI) is a measure of the uniqueness of the indexing of the Kikuchi patterns and provides an indication of the quality of the EBSD pattern. The CI evaluates the degree of agreement between the measured pattern and the reference database, and can also be utilized to differentiate between bainite and ferrite [28].

Bainite and martensite can be distinguished in principle by examining these metrics that evaluate the quality of diffraction patterns. The average IQ and CI tends to be higher for bainite due to a lower concentration of defects. Figure 2.8 displays the distribution of average CI for each object in both sets of samples.

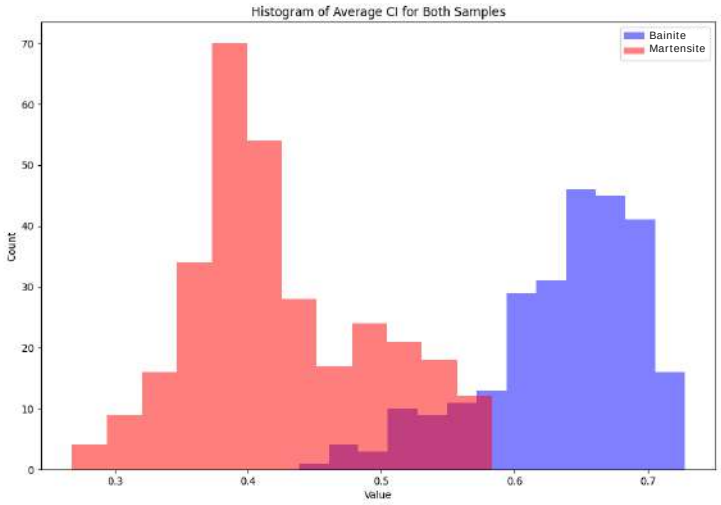

Figure 2.8: Distribution of average CI for every object with compositions A and C.

To address the limitations of methods that use diffraction pattern quality metrics, alternative EBSD maps based on misorientation relationships have been proposed [2, 32]. Local misorientations can be characterized using several pa-

rameters. The Grain Orientation Spread (GOS) represents the average deviation in orientation between each point in a grain and the grain's average orientation [31]. The Grain Average Misorientation (GAM) is a comparable metric that calculates the average misorientation between neighboring measurement points within a grain. The Kernel Average Misorientation (KAM) is similar to GAM but is computed within a defined kernel of varying sizes. In this approach, the misorientations between all neighboring points within the kernel are averaged [31]. These parameters can be used to draw conclusions about local dislocation densities and distinguish between microconstituents.

It is important to highlight that grain-based parameters are extremely sensitive to the grain definition, which is why kernel-based parameters are often preferred. However, since the classification in this study is object-based, it is worthwhile to consider GOS and GAM as relevant parameters. Additionally, it is worth noting that the grain definition for GOS and GAM differs from the parent austenite grain (PAG) definition used in this work for object extraction.

Considering these parameters, a manual system for characterizing different steel phases is theoretically possible. However, it would be very laborious and time-consuming. By applying machine learning systems, the model can identify relevant patterns in the data, achieve an objective and reproducible characterization, and even surpass the efficiency of manual systems in a shorter time.

Correlative microscopy is another modern approach to microstructural analysis that enables the combination of different characterization methods to overcome the limitations of each individual method [33]. Due to the different physical interactions and detectors, features from various length scales and complementary information sources can be combined, providing a more comprehensive understanding of the sample's microstructure [5].

The first step in correlative microscopy is to generate images of the same sample areas using different imaging techniques, such as LOM, SEM, and EBSD. However, due to the different underlying physics of the image generation in these techniques, simply overlaying images by translation, rotation, and scaling is not possible. Additionally, acquisition conditions like the tilt angle or rotation can make it difficult to align images [33]. To address this issue, image registration is employed to align two or more array-shaped image data, typically images of the same object captured at different times, angles, and/or using different sensors

[34].

Image registration involves several steps. The first step is feature detection, which involves identifying distinct objects in each image. In the feature matching step, identical data points in the two images are related to each other. A transformation model is then estimated, which involves determining the mapping function and its parameters. Finally, the source image is resampled and transformed based on the estimated mapping function and parameters [34].

One way to implement the image registration algorithm is by using the open-source software Fiji [7]. The feature extraction step can be completed automatically using the SIFT (Scale-Invariant Feature Transform) plugin [35]. However, in the case of the images used in this work, the SIFT algorithm could not detect features due to the presence of fine microstructural features. Therefore, manual feature extraction had to be performed. To perform image registration, the bUnwarpJ plugin [8] can be utilized, which uses elastic deformation to align the images.

This technique can be applied to any source of information that generates a large number of possible combinations. This allows for a more comprehensive and unbiased characterization. Moreover, by registering LOM images to the IQ map, it becomes possible to overlay other EBSD maps onto the LOM image [36]. Figure 2.9 illustrates some of the potential ways to combine different sources of information.

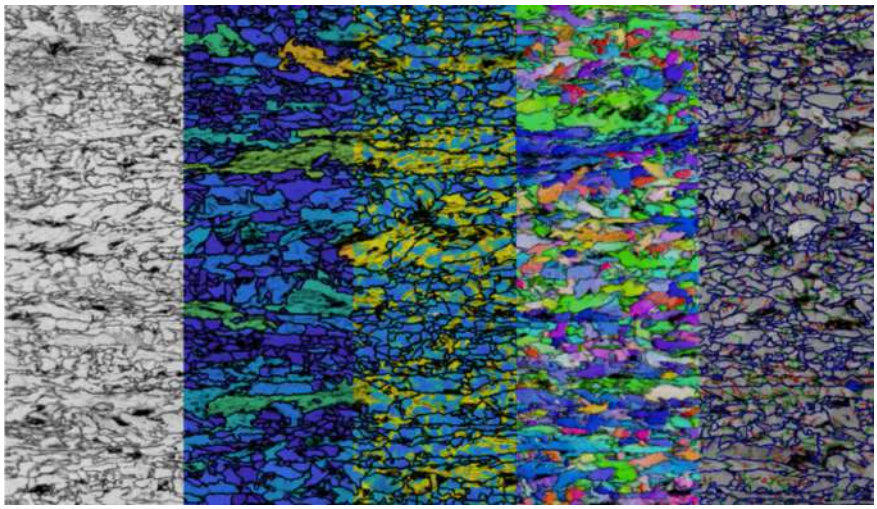

Figure 2.9: An overlay of a selection of EBSD maps with the Nital-etched LSM image registered on it. From left to right: LSM - GAM - KAM - IPF - boundaries based on the relative rotation angles of the crystallites to each other (red: 2-5°, green: 5-15°, blue: greater than 15°) [37].

2.2 Fundamentals of Machine Learning

2.2.1 Introduction

Artificial Intelligence (AI) is a rapidly growing field that has gained significant attention in recent years. Essentially, AI refers to the study of agents, such as computer systems, that can perform actions based on perceptions from the environment [38]. These systems have the ability to learn and complete tasks that previously required human intelligence.

Machine learning is a field of artificial intelligence that enables computer systems to learn and improve their performance on a task without being explicitly programmed [39]. Instead, these algorithms allow the computer to identify patterns and relationships in data by analyzing large amounts of information and using statistical models. Through this process, the system can make predictions or decisions based on the patterns it has learned. To provide a thorough understanding of the methodologies employed in this research, the following

section will provide a detailed explanation of machine learning approaches.

The growth and impact of AI and ML are reflected in the increasing number of publications per year in these fields. The 2022 AI Index report [40], which tracks the progress of AI research and development, shows a significant rise in the number of publications on these fields in recent years, as depicted in Figures 2.10 and 2.11. This growth is attributable to various factors, such as the availability of big data, improvements in computer processing power, and advances in algorithm development. For instance, the report states that the cost of training an image classification system has decreased by 63.6%, while training times have improved by 94.4% since 2018. Another noteworthy aspect is the diagnostic capabilities of AI algorithms, which can match or even surpass those of expert doctors for numerous conditions, particularly those that rely on image analysis [41]. As a result of these advances, AI is expanding into a wide range of fields, including materials science.

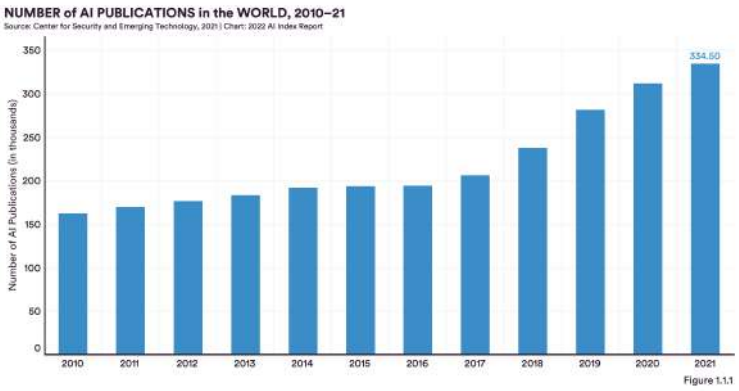

Figure 2.10: Number of AI publications in the world between 2010 and 2021.

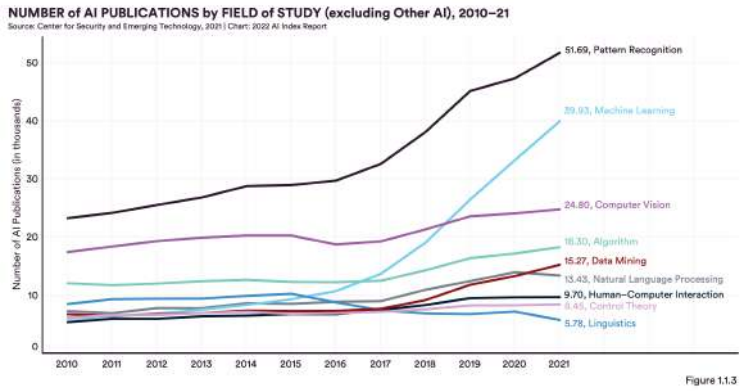

Figure 2.11: Number of AI publications by field of study in the world between 2010 and 2021.

Numerous studies have demonstrated the effectiveness of various methods in accurately, objectively, and reproducibly characterizing complex steels. For instance, support vector machine (SVM) models have been utilized in some studies to classify the microstructures of steels based on morphological parameters [4, 42]. Müller et al. [43] employed a combination of textural parameters, including the Haralick parameter and the Local Binary Pattern, to classify pearlite, martensite, and four types of bainite using SVM. Tsutsui et al. [44] used crystallographic features obtained by EBSD to classify martensite and bainite using both SVM and Random Forest. Zaefferer et al. used the Kernel Average Misorientation (KAM) to identify and determine the volume fraction and localization of bainite in a low-alloyed TRIP [45]. Martinez Ostormujof et al. used an U-Net architecture to perform semantic segmentation of martensite and ferrite with EBSD maps [30].Another study employed a Fully Convolutional Neural Network (FCN) to perform pixel-wise segmentation of multiple phases in steel microstructures [6].

2.2.2 Machine Learning

As mentioned earlier, machine learning is a subfield of artificial intelligence, in which computers can learn to solve complex tasks without being explicitly programmed to do so [39]. The fundamental premise of machine learning is to use statistical models to identify patterns in data that may not be readily

apparent to humans. To perform a machine learning task, the first step is to select an appropriate algorithm that can perform the analysis. Then, a set of hyperparameters that regulate the learning process are chosen, and the data is supplied to the algorithm for analysis. The computer uses an iterative, statistical, trial-and-error process to decipher the patterns present in the data, ultimately learning to make accurate predictions for new data based on what it has learned from the training data.

Machine learning can be classified into three broad categories based on the type of feedback that accompanies the inputs and thus determines the type of learning. The first category is supervised learning, where the inputs are accompanied by so-called ground truths or labels, and the model is trained to learn the function that maps the inputs to the ground truths [46].

The second category is unsupervised learning, where the computer learns to decipher patterns in the data without explicit feedback. Clustering is a common example of unsupervised learning, where the computer analyzes the features or attributes of the data and aims to identify groups or clusters that share similar characteristics, without any prior knowledge of the class definition.

The third category is reinforcement learning, where the agent learns through punishments or rewards. Reinforcement learning is typically used in scenarios where the model interacts with the environment and learns to take actions that maximize a reward [47].

In this work, the focus is on supervised learning, as the goal is to train a model to classify bainite and martensite objects. Therefore, the machine learning algorithm will use labeled data consisting of EBSD matrices for each object, and attempt to learn the function that maps the input data to the correct classification of either bainite or martensite.

Supervised learning problems can be divided into two main categories: classification and regression. In a classification problem, the output is one of a finite set of values, such as bainite or martensite. On the other hand, in a regression problem, the output is a continuous number [46]. In material science, image segmentation is another commonly employed supervised learning task that involves pixel-wise classification. The goal of image segmentation is to divide an image into fragments and assign them labels, enabling the identification of distinct regions or objects within the image.

More formally, the task solved by supervised learning is that, given a training set of N examples input-output pair:

$$(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$$

where each pair was generated by the unknown function $y = f(x)$, discover the function h that better approximates the true function f. The function h is called the hypothesis about the world and we can say that is the model of the data [38]. The outputs y_i are the ground truth.

The effectiveness of a hypothesis is not determined by its performance on the training set alone, but rather by its ability to handle unseen inputs. To evaluate its performance on new data, we need to use a separate set of input-output pairs known as the test set. A hypothesis is said to generalize well if it can accurately predict the outputs of unseen data. In other words, a hypothesis that performs well on both the training and test sets is more likely to be a good model of the underlying function than a hypothesis that only performs well on the training set.

A model's ability to generalize well is affected by two important factors: bias and variance. Bias refers to the tendency of a predictive hypothesis to deviate from the expected value when averaged over different training sets . In other words, it is the difference between the expected predictions of the model and the true values in the data. High bias models are typically too simplistic and unable to capture the complexity of the data, resulting in underfitting. On the other hand, variance refers to the variability in the model's predictions when it is trained on different subsets of the data [38, 48]. Models with high variance are often too complex and overfit to the training data, leading to poor performance on new data. Balancing the bias-variance tradeoff is a key challenge in machine learning, as models with low bias and low variance are more likely to generalize well to new, unseen data.

All machine learning algorithms aim to minimize a loss function in order to learn. The loss function evaluates the discrepancy between the predicted output and the actual output for each example in the training data. The objective of the algorithm is to optimize the model's parameters so as to minimize this function. Therefore, the best hypothesis or model is the one with the lowest loss, as it exhibits the least difference between its predictions and the actual values.

The hypothesis h is characterized by a set of parameters W, and the type of model determines the number of parameters and their relationship to h. The goal of the learning algorithm is to minimize the loss function, which is achieved using the gradient descent algorithm. To begin training, we initialize the parameters W, then calculate an estimate of the gradient and move in the direction of the steepest descent, repeating the process until the model converges to a point in weight space with a minimum loss value. The gradient descent algorithm can be expressed as follows [38]:

for w_i in W do:

$$w_i = w_i - \alpha \frac{\partial Loss(W)}{\partial w_i} \tag{2.1}$$

Here, α represents the learning rate, which decides the step size that determines how much each iteration adjusts the parameter values. This hyper-parameter can be set as a fixed constant or can be decayed over time as the learning process proceeds. The choice of learning rate affects the convergence of the algorithm and can significantly impact the performance of the model. A learning rate that is too high can lead to overshooting and divergence, while a learning rate that is too low can result in slow convergence and suboptimal solutions. Therefore, selecting an appropriate learning rate is a vital component in the training process of a machine learning model.

2.2.3 Support Vector Machine

Support Vector Machine (SVM) is a popular machine learning algorithm used for classification and regression analysis. Although their popularity has decreased with the emergence of deep learning networks and random forests, SVMs are still widely used in many applications. SVMs are particularly useful when the available data has a clear separation between classes, or when the goal is to identify a subset of data points that are important for classification [49].

SVMs are designed to construct a maximum margin separator [38], which is a decision boundary that has the largest possible distance to example points. The idea is to choose a decision boundary that will generalize well, which means it will perform well on new data that is not part of the training set. The SVM algorithm finds the maximum margin separator by solving an optimization

problem that seeks to maximize the distance between the decision boundary and the closest example points. Figure 2.12 provides a clear illustration of the difference between SVM and other models like linear regression, highlighting how SVM's unique approach can lead to improved generalization performance.

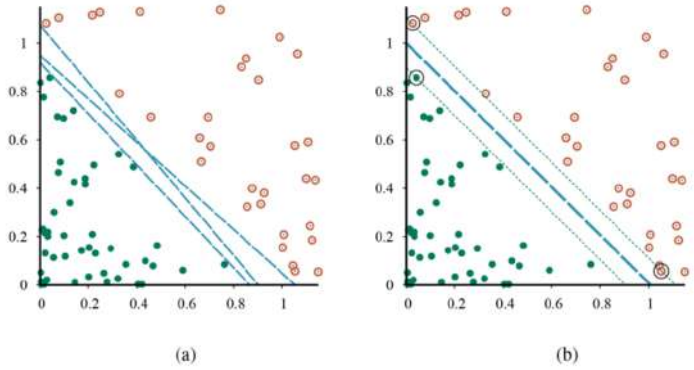

Figure 2.12: Support vector machine classification: (a) Two classes of points and three candidate linear separators. (b) The maximum margin separator (heavy line), is at the midpoint of the margin (area between dashed lines). The support vectors (points with large black circles) are the examples closest to the separator; here there are three. [38]

One of the attractive features of SVMs is that they can create a linear separating hyperplane, but they also have the ability to embed the data into a higher-dimensional space, using the so-called kernel trick [38, 49]. This means that data that are not linearly separable in the original input space can often be easily separated in the higher-dimensional space. The kernel function computes the inner product of two points in the higher-dimensional space without actually computing the transformation [49]. Figure 2.13 shows an example of how the kernel trick can be used to find a non linear decision boundary. There are several types of kernel functions, such as the linear kernel, polynomial kernel, and Gaussian (or radial basis function) kernel, among others. The SVM classifier employed in this work utilizes the radial basis function as its kernel.

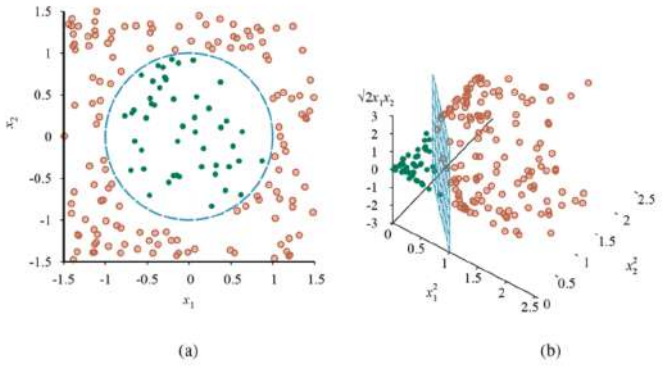

Figure 2.13: (a) A two-dimensional training set with a circular decision boundary (b) The same data after mapping into a three-dimensional input space. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions [38]

SVMs are nonparametric, which means that the separating hyperplane is defined by a set of example points, not by a collection of parameter values [50]. The SVM model keeps only the examples that are closest to the separating plane, which are called support vectors. This property of SVMs allows them to be flexible enough to represent complex functions while also being resistant to overfitting.

In summary, SVMs are a powerful supervised learning model that can create a maximum margin separator to generalize well on new data, create linear separating hyperplanes, and embed the data into a higher-dimensional space using kernel functions. SVMs are nonparametric and resistant to overfitting, while still being flexible enough to represent complex functions.

2.2.4 Random Forest

Random Forest is a powerful ensemble learning algorithm that combines multiple decision trees to make more accurate predictions. It is a type of supervised learning algorithm that can be used for both regression and classification problems.

Ensemble learning is a machine learning approach that involves selecting

multiple hypotheses, known as base models, and combining their predictions through techniques such as averaging, voting, or other machine learning algorithms [38]. This approach is used to reduce both bias and variance. Ensemble models can be more expressive and have lower bias than individual base models with restrictive hypothesis spaces. Additionally, by combining multiple classifiers, it is less likely to misclassify new examples, which helps to reduce variance [38].

A decision tree is a model that recursively divides the data into subsets based on the most significant features or attributes. The tree is constructed by starting with a single node, called the root, that represents the entire dataset. Then, at each internal node, the algorithm selects a feature to split the data into two or more subsets, based on a particular criterion such as entropy, Gini impurity, or information gain. This process is repeated for each subset, creating a tree-like structure with branches representing different paths or decision rules. At the leaves of the tree, the final prediction or decision is made.

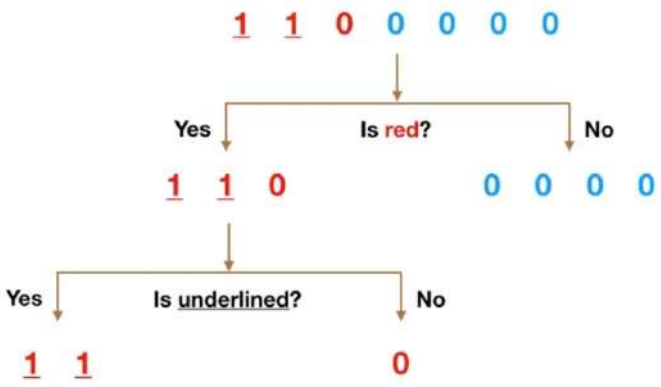

Figure 2.14: Simple example of a decision tree to illustrate the working principle of decision trees in a straightforward manner. [51]

Ideally, each decision tree in a Random Forest should be independent, as this would maximize the effectiveness of the ensemble learning technique. In practice, it is not possible to achieve complete independence between trees, as they share some of the same data and assumptions [38]. However, the Random Forest algorithm can introduce some degree of independence by using a technique called bagging (Bootstrap Aggregation).

The bagging technique takes advantage of the fact that decision trees are very sensitive to the data with which they are trained. Small changes in the training set can result in significantly different tree structures. Bagging allows each individual tree to take random samples from the data set with replacement, resulting in different trees. For example, if we had the following training data consisting of the grain sizes of a steel in micrometers (30, 35, 40, 45, 50, 55), one of our decision trees in a Random Forest model might receive the following list (30, 35, 35, 40, 40, 55), where some sizes are repeated due to the sampling with replacement. This variation in the training sets used to create the trees helps to reduce overfitting and improve the accuracy of the overall Random Forest model.

In addition to bagging, Random Forest has several hyperparameters that can be adjusted to optimize performance. But the most important and the one used in this work is the number of trees. It has been demonstrated that as the number of trees increases, the error function converges. Nevertheless, it is important to note that this convergence does not imply that the error tends to zero as the number of trees approaches infinity [52].

Overall, Random Forest has several advantages, including its ability to handle high-dimensional data, detect feature importance, and handle missing values. However, it can be computationally expensive and may not perform as well in task like image classification. This is because Random Forest treats each feature independently and makes decisions based on individual feature thresholds, which may not adequately capture the spatial dependencies present in image data.

2.2.5 Artificial Neural Network and Deep Learning

Artificial Neural Networks (ANNs) are a type of machine learning model that is designed to resemble the structure and function of biological neural networks. ANNs are composed of interconnected computational units called neurons, which transform an input signal into an output signal using a mathematical function. The output signals are then propagated to the next layer of neurons, and the process repeats until the network reaches its final layer [53].

Each neuron in an artificial neural network computes a linear combination of the inputs with the corresponding weights, which is then passed through

a nonlinear function called the activation function. This function is an essential component of ANNs, as it allows the network to represent complex, nonlinear relationships. If the network were composed only of neurons that applied linear combinations of inputs, it could be reduced to a single neuron due to the superposition principle, consequently limiting its ability to model only linear functions [38].

The mathematical computation made by each neuron can be formally expressed as:

$$a_j = g_j(\sum_i w_{i,j} a_i)$$

Here, a_j represents the output of the neuron j, $w_{i,j}$ represents the weight of the connection between neuron i and neuron j, a_i represents the output of neuron i, and g_j represents the activation function associated with neuron j [38].

Rectified Linear Unit (ReLU) and Softmax are two common activation functions used in neural networks and are the ones used in this work. ReLU is a simple nonlinear function that is defined as $g(x) = \max(0, x)$, which means that it outputs the input value if it is positive, and zero otherwise. ReLU is a popular choice because it is computationally efficient and easy to optimize. Softmax is another common activation function used in the output layer of neural networks for classification tasks. Softmax normalizes the outputs of a layer so that they sum to one, which makes it suitable for predicting the probability of each class. Softmax is defined as [38]:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Here, z_i is the input to the activation function for class i, and K is the total number of classes

ANNs are composed of at least two layers of neurons: an input layer and an output layer. Additional layers between the input and output layers are known as hidden layers, and ANNs with multiple hidden layers are called deep neural networks (DNN). By combining multiple units into a network, a complex function can be created that captures the complexity of real-world data. At each layer of the network, the values computed represent a different representation of the input. Each layer transforms the representation produced by the preceding

layer to create a new representation. As a result, deep networks often uncover meaningful intermediate representations of the data. However, the meaning of internal layers in deep networks may sometimes be unclear to humans, even if the output is correct [38].

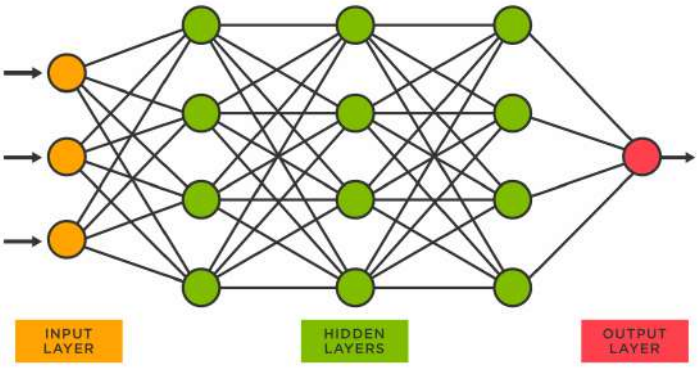

Figure 2.15: Schematic representation of a deep neural network with 3 hidden layers. [54]

Deep neural networks learn by using the gradient descent algorithm, as explained in Equation 2.1. However, for multilayer networks, the loss function is a complex function of the weights and activations of the previous layers. Therefore, the backpropagation algorithm is used to calculate the derivatives of the loss function with respect to each of the weights by applying the chain rule. These calculated gradients are then used to update the parameters.

Given that DNNs typically have a large number of parameters, they require a considerable amount of data to effectively learn the features of the data. Data augmentation addresses this need by artificially expanding the training dataset through the application of various transformations and modifications to the existing data samples. By introducing random modifications such as rotations, translations, scalings, and distortions to the original data, data augmentation significantly improves the diversity and variability within the training set. As a result, the utilization of data augmentation leads to improved generalization and robustness of DNNs, enabling them to better handle various real-world scenarios and improve overall performance. Data augmentation is also used in CNNs.

The methods discussed so far (SVM, RF and ANNs) take vectors with a given number of attributes as input and are not explicitly designed to handle image inputs. Although it is possible to convert an RGB (red, green, blue) image of size n by m pixels into a flattened 1D vector with $3 \times n \times m$ attributes and use these methods, this approach overlooks the fact that the RGB triplets belong to the same pixel and the significance of pixel adjacency in image data. As a result, applying these methods to vectorized images can lead to a loss of essential spatial information and, consequently, reduce accuracy [38]. Furthermore, using a flattened vector of an image results in a large number of attributes, which leads to an extremely high number of parameters in a conventional neural network. This vast parameter space requires correspondingly vast numbers of training images and a huge computational power to run the training algorithm [38].

2.2.6 Convolutional Neural Network

Convolutional neural networks (CNNs) are used to address the limitations of previous methods in image manipulation by learning and extracting relevant features from images or grid-structured data that exhibit some kind of relationship between adjacent pixels. CNNs work in a similar way to traditional neural networks, but have special layers that apply operations that take into account the spatial relationship between inputs. These layers have carefully designed connections to extract important features from the inputs while reducing the number of parameters to be learned [53]. CNNs are typically composed of various types of layers including convolutional layers, pooling layers, activation layers, and fully connected layers used for classification.

The convolution layer is the fundamental building block of a CNN and is responsible for extracting features from an input image. The convolution operation involves applying a filter or kernel to every pixel in the input image, and computing the dot product between the filter parameters and the matching grid in the input volume. The parameters of the filter are shared across the entire convolution, which is sensible since a particular shape present in any part of the image should be processed in the same way, regardless of its specific spatial location. This also helps reduce the number of parameters. Typically, the height and width of the filter in a convolutional layer are smaller than those of the layer input, while its depth is the same as that of the input [53]. The resulting grid after the convolution is called a feature map. An example of how

the convolution is applied is shown in Figure 2.16.

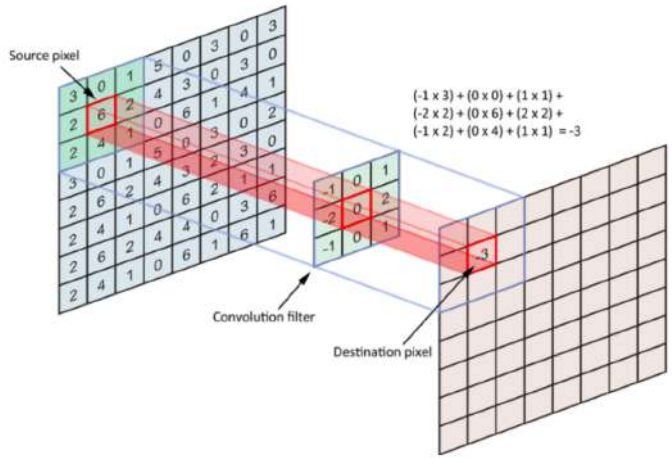

Figure 2.16: An example of convolution operation is shown, where a 3×3 filter is applied to a depth 1 input (such as a black and white image) to extract features and produce a feature map. [55]

In order to extract a diverse set of features from the input image, it is common to use multiple filters in each convolutional layer, as each filter captures a distinct aspect of the image. The number of filters used in each layer directly affects the model's capacity, as it determines the number of learnable parameters [53]. Furthermore, increasing the number of filters results in more feature maps being produced in subsequent layers. Typically, as we move to deeper layers, the spatial dimensions of the feature maps decrease due to pooling while the depth in terms of the number of feature maps increases.

Pooling layers are used to reduce the spatial dimensions of the feature maps produced by the convolutional layers. This helps reduce the number of parameters, which can lower computation time, save memory, and prevent overfitting [56]. Max pooling is the most common pooling operation, which involves taking the maximum value within a small neighborhood of the feature map and outputting it as a single value in the output map. By doing this, the spatial resolution of the feature map is effectively reduced by a factor of the pooling size. Figure 2.17 demonstrates how pooling is applied.

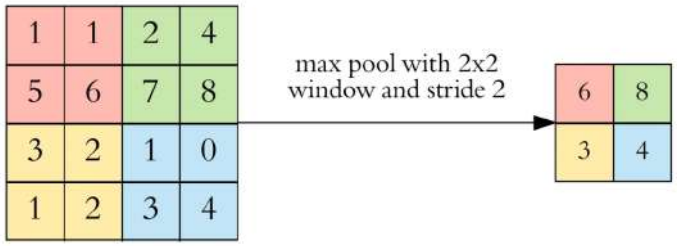

Figure 2.17: Example of max pooling applied with a 2x2 window and a stride of 2. With a stride of 2, the window is moved 2 pixels for each step. [56]

In a CNNs, the activation layer typically uses the ReLU function just like in a regular neural network. The ReLU activation function is applied to each value in a layer to create thresholded values that are then passed to the next layer [53]. Unlike pooling and convolution operations, applying the ReLU function does not alter the dimensions of a layer, as it is a one-to-one mapping of activation values.

After sufficiently reducing the dimensions of the input, the resulting output is flattened and typically fed into one or more fully connected layers, which perform the final classification of the image. These layers are similar to the ones used in a traditional neural network, and they typically use a softmax activation function to produce a probability distribution over the possible classes. An example of a typical architecture of a CNN is shown in Figure 2.18.

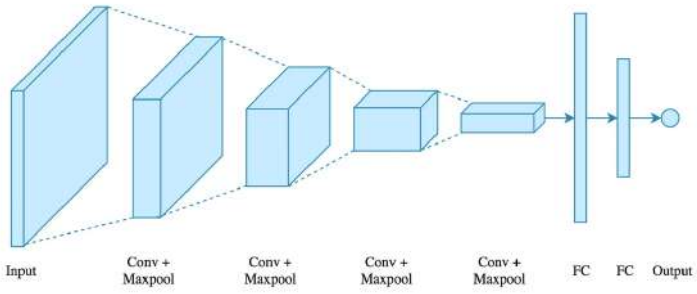

Figure 2.18: Example of a CNN architecture. The figure shows how the spatial dimensions of the input are reduced after applying convolutional and pooling layers, and how the depth of the feature maps increases. Fully connected layers are used at the end to make predictions. A Softmax activation function is employed at the end of the network to generate a probability distribution across the possible classes. [56]

2.2.7 K-Fold Cross-Validation

K-fold cross-validation is a statistical technique for evaluating the performance of machine learning models. It involves dividing the dataset into k equal-sized subsets, or "folds," where k is usually a number between 5 and 10. For each fold, the model is trained on the remaining k-1 folds and evaluated on the validation set. The process is repeated for each fold, and the average performance metric across all k folds is computed to estimate the model's performance.

K-fold cross-validation has two primary benefits. Firstly, it provides a more unbiased estimate of model performance on new, unseen data compared to a simple train/test split [57]. Secondly, k-fold cross-validation is particularly useful when the dataset is small because it allows each example to be used for both training and validation, but not at the same time [38]. This ensures that all the data is utilized effectively to train and evaluate the model. Additionally, k-fold cross-validation can also be used to compare the performance of different models or hyperparameters, making it a valuable tool for testing and comparing models.

Chapter 3

Experimental Procedure

3.1 Sample Preparation

The test samples used in this work were supplied by Aktien-Gesellschaft der Dillinger Hüttenwerke and consist of dilatometer samples of known chemical composition and heat treatment. The samples were divided into groups based on their chemical composition, and each group was assigned a letter (A, B, C, etc.). Within each group, the samples were numbered according to their cooling rate, with lower numbers indicating higher cooling rates and higher numbers indicating lower cooling rates. For the purposes of this work, only the A and C steel samples were used. Their composition can be seen in Table 3.1. Aditionally, the cooling rates for each A and C samples can be seen in Tables 3.2 and 3.3

Obtaining reproducible micrographs is essential for this work for several reasons. First, the images obtained by the different methods must be of high quality to ensure that correlative microscopy can be performed effectively. Any scratches, impurities, or etching artifacts could hinder the procedure and compromise the accuracy of the results. Secondly, it is important to ensure that the different procedures do not alter the assignment of ground truths, as this could affect the validity of the results. Finally, we must take care to avoid allowing artificial intelligence models to learn features that are not inherent to the material itself, but are instead related to the processing. By obtaining high-quality,

	A	C
Name		
C	0.088	0.219
	Compositions censored by agreement with Aktien-Gesellschaft der Dillinger Hüttenwerke.	

Table 3.1: Composition of samples A and C.

Sample	Cooling Rate (in °C / s)
A1	3600
A2	2400
A7	56.25
A9	15
A11	5

Table 3.2: Cooling rate (in °C / s) for every A sample.

Sample	Cooling Rate (in °C / s)
C1	18000
C3	1800
C5	360
C8	45
C10	11.25

Table 3.3: Cooling rate (in °C / s) for every C sample.

Abrasive Material	Duration
180 SiC (75 µm approx.)	1 min
320 SiC (46 µm approx.)	1 min
600 SiC (26 µm approx.)	1 min
1200 SiC (14 µm approx.)	2 min
6 µm Struers DiaDuo 2	6 min
3 µm Struers DiaDuo 2	3 min
1 µm Struers DiaDuo 2	5 min
OP-S 0.05 µm	2 x 2 min

Table 3.4: Metallographic preparation steps.

reproducible micrographs and carefully controlling the factors that could affect the results, we can ensure the accuracy and reliability of our findings.

The samples were first mounted on a conductive powder suitable for use in the scanning electron microscope. They were then ground and polished in accordance with the steps outlined in Table 3.4. The grinding was performed using the wet method, with the specimens being rotated 90° after each step until the marks of the previous grit had completely disappeared. This typically took the amount of time mentioned in Table 3.4. The polishing was then carried out automatically by counter-rotating the samples to remove any unevenness resulting from the grinding process.

After each grinding and polishing step, the specimens were cleaned with absorbent cotton under running water to remove any residues and metal dust. Subsequent rinsing with ethanol and drying with compressed air completed the respective preparation step.

Correlative microscopy involves taking micrographs of the same area of a specimen using different methods to gain more information about the microstructure. In order to capture images of the same area using different microscopy techniques, it is necessary to clearly mark a region of interest (ROI) on each specimen. For this purpose, a pattern of indentations was used to define

the ROI, covering an area of 500 um x 500 um on each sample. The patern can be seen in Figure 3.1.

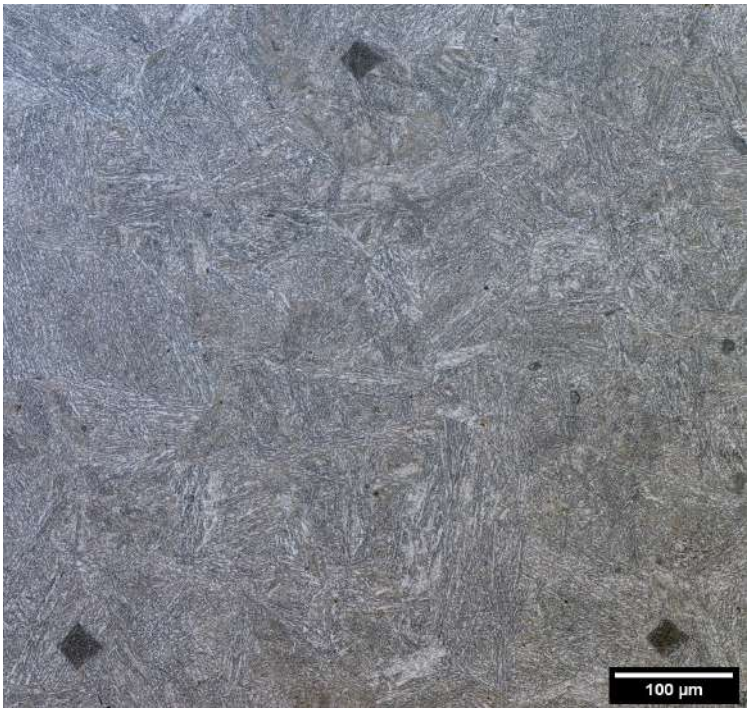

Figure 3.1: Region of interest (ROI) marked with a pattern of indentations on a light optical microscope image of sample C10.

After marking the ROIs and polishing the samples up to the OP-S, EBSD measurements were conducted on the ROI of each sample. Once the EBSD analysis was successfully completed, the samples were repolished using a 1 µm abrasive and then etched with Nital for approximately 30 seconds. Finally, optical microscopy (LOM) and scanning electron microscopy (SEM) images were taken after the etching process.

3.2 Experimental Methods

3.2.1 Electron Backscatter Diffraction

EBSD measurements were conducted using a Zeiss Merlin equipped with an EDAX detector. Data was processed and exported using OIM software. The grain definition was based on a misorientation threshold of 5°, in compliance with both the internal convention of Aktien-Gesellschaft der Dillinger Hüttenwerke and the ASTM E2627 standard for EBSD grain size determination [58].

To ensure accurate results, EBSD requires a flat surface. Thus, polishing with OP-S was required. Therefore, polishing with OP-S was necessary. The specimens were polished twice for two minutes each, as specified in Table 3.4, using a colloidal silica suspension (OP-S) and water. The suspended oxide particles have a size smaller than 0.05 µm, resulting in a surface with irregularities of comparable magnitude.

3.2.2 Light Optical Microscope

The micrographs were obtained using an Olympus OLS 4100 LSM microscope and the Olympus LEXT software. Both optical microscope and laser scanning microscope (LSM) images were obtained. Each image has a resolution of 1024 pixels x 1024 pixels. At 1000x magnification, a sample area of 130 µm x 130 µm is imaged. In order to obtain a micrograph of the entire ROI, 25 pictures were taken in a 5×5 grid and stitched together using the Microsoft Image Composite Editor program. Parameters like brightness and contrast were kept constant in order to get reproducible micrographies.

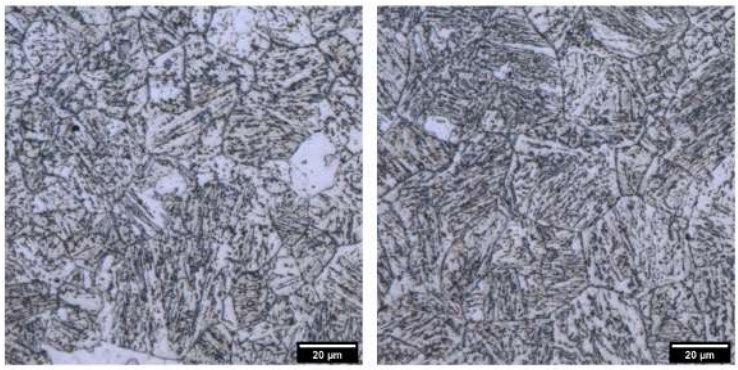

Figure 3.2: Two individual sections of the ROI of sample A06 taken with LOM.

3.2.3 Scanning Electron Microscope

Scanning electron microscopy images were obtained using a Zeiss Supra microscope at 1000x magnification. The operating voltage was set to 5 kV, with a beam current of 10 nA and a working distance of 5 mm. The micrographs were reconstructed using a combination of two detectors: a secondary electron detector, which provided 70% of the signal and enhanced topographical information, and an InLens detector, which contributed the remaining 30% of the signal and enhanced contrast between different phases. This combination resulted in high-resolution images with detailed information on both topography and phase difference, facilitating the accurate registration of images and identification of phases.

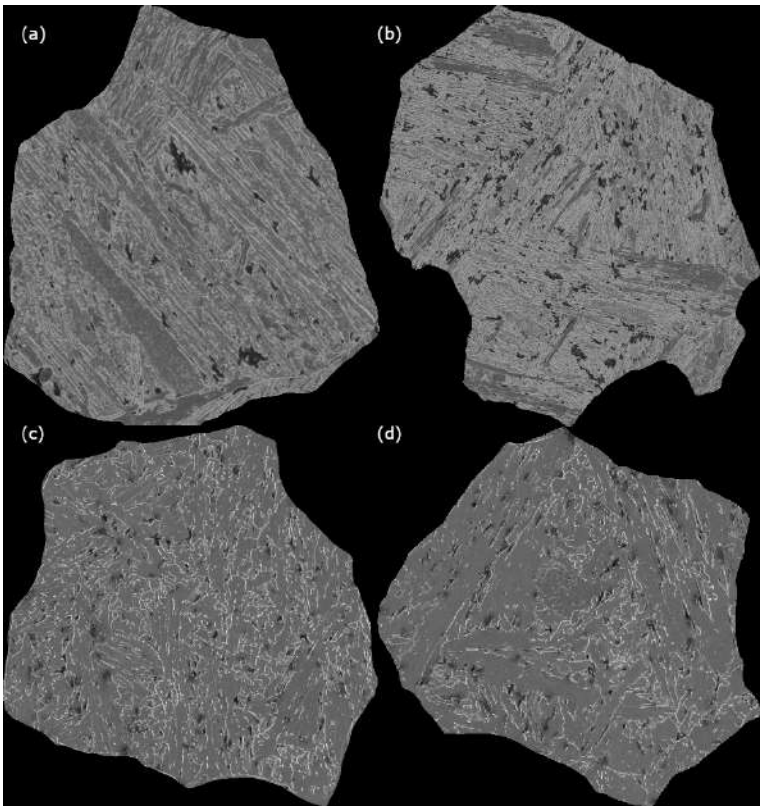

Figure 3.3: SEM image of objects: (a) 21 of sample A2 labeled as martensite, (b) 92 of sample A2 labeled as martensite, (c) 31 of sample A7 labeled as bainite, (d) 64 of sample A7 labeled as bainite.

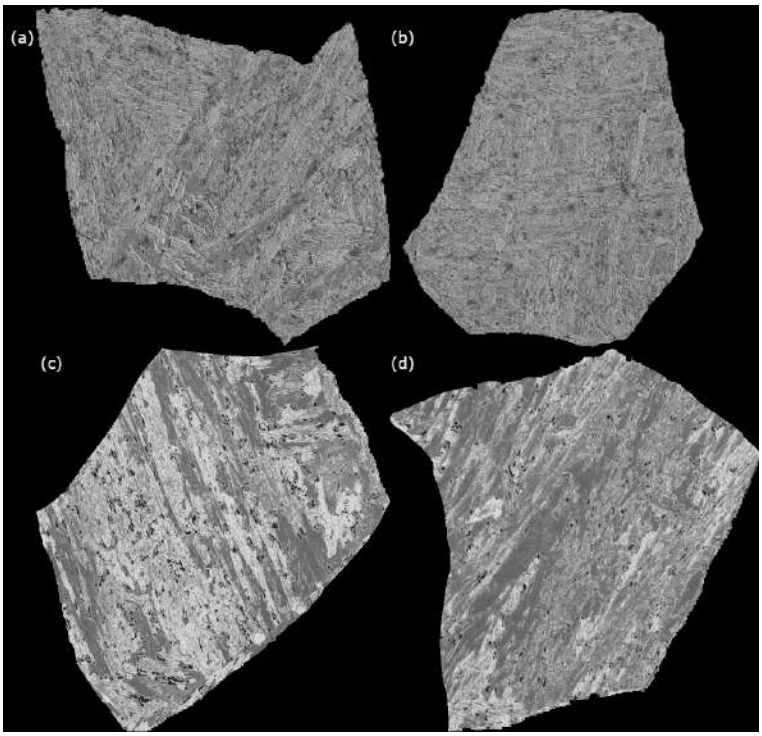

Figure 3.4: SEM image of objects: (a) 18 of sample C1 labeled as martensite, (b) 19 of sample C1 labeled as martensite, (c) 10 of sample C8 labeled as bainite, (d) 24 of sample C8 labeled as bainite.

3.3 Data Processing

3.3.1 Post Processing of EBSD Data

After acquiring the EBSD data, the software provides a one-dimensional list of all the data. However, this data needs to be converted into two-dimensional matrices in order be registered with the other images. A challenge arises due to the format of the EBSD data, which is typically acquired in a hexagonal grid. Since the pixels in the images are organized in a rectangular grid, the EBSD data must be reformatted. In order to transform the data from the hexagonal to the rectangular grid, interpolation is required because the step sizes in the hexagonal grid are not uniform. These challenges are addressed

by implementing a MATLAB script based on the MTEX [59] plugin, which is specifically designed for processing EBSD data.

The output of this script is a $1320 \times 1320 \times 6$ array containing the normalized EBSD data of the ROI. The third dimension of the array corresponds to the 6 EBSD maps: IQ, CI, KAM1, KAM3, GOS and GAM, respectively.

3.3.2 Registration

LOM images were registered to the EBSD IQ map using the open-source software Fiji [7]. Feature detection was performed manually since SIFT extraction did not yield satisfactory results. This involved selecting distinctive features and correlating them between the two images. Once the features were extracted, registration was carried out using the bUnwarpJ plugin [8].

Subsequently, SEM images were registered to LOM using the same methodology. The registration of LOM images to the IQ map enabled the overlaying of the other EBSD maps. Finally, with the SEM images registered to the LOM images, an overlay was constructed for each sample, incorporating LOM and SEM images, as well as the EBSD maps: IQ, CI, KAM1, KAM3, GOS, and GAM.

3.3.3 Prior Austenite Grain Boundary Mask

To extract individual objects for ML model training and classification, a mask of the Prior Austenite Grain Boundaries (PAGB) [9] was manually created. The mask was drawn using the open-source software GIMP [60] on the overlay of optical images and EBSD maps, allowing for the identification and reconstruction of these boundaries. Figure 3.5 illustrates an example of this mask for the ROI of sample A01.

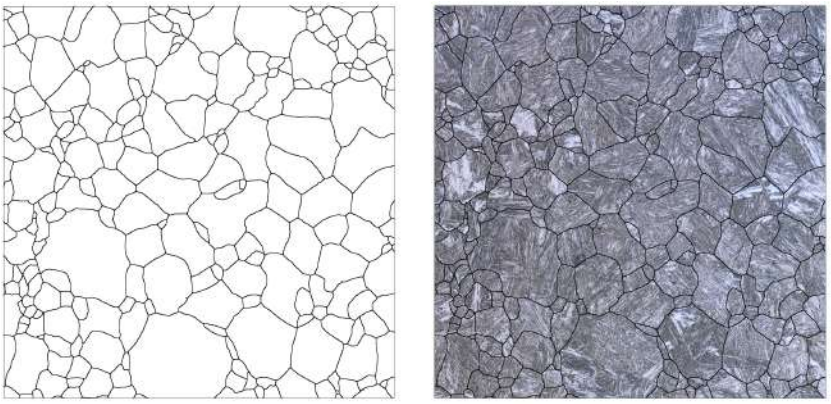

Figure 3.5: Left: Grain boundary mask of the ROI of sample A01. Right: LOM image of the ROI of sample A01 with superimposed grain boundary mask.

3.3.4 Obtaining the objects

The objects used for training the ML model consist of arrays of six EBSD maps: IQ, CI, KAM1, KAM3, GOS, and GAM. To obtain such objects, a series of Python functions must be implemented on the EBSD array.

First, with the grain boundary mask, masks for each object are created. These masks consist of black and white images with the same dimensions as the overlay of registered images. The mask is a completely black image, except for the particle object where it is white. The Python function responsible for this is included in the Appendix Source Code 1. This approach utilizes the OpenCV module [61], using the "findContours" function to identify each object from the grain boundary mask. These contours are then drawn on a black image and filled with white, and each image is saved.

The grain masks are then applied to the EBSD array to obtain arrays of each object with the six EBSD maps. To achieve this, the EBSD matrix is loaded using the Python module SciPy [62]. The objects are then cropped to the minimum height and width possible, and the resulting images are saved. The Python function that implemnts this is included in the Appendix Source Code 2.

Figure 3.6 displays the six EBSD maps of object 49 in sample C01, while Figure 3.7 shows the corresponding object matrix.

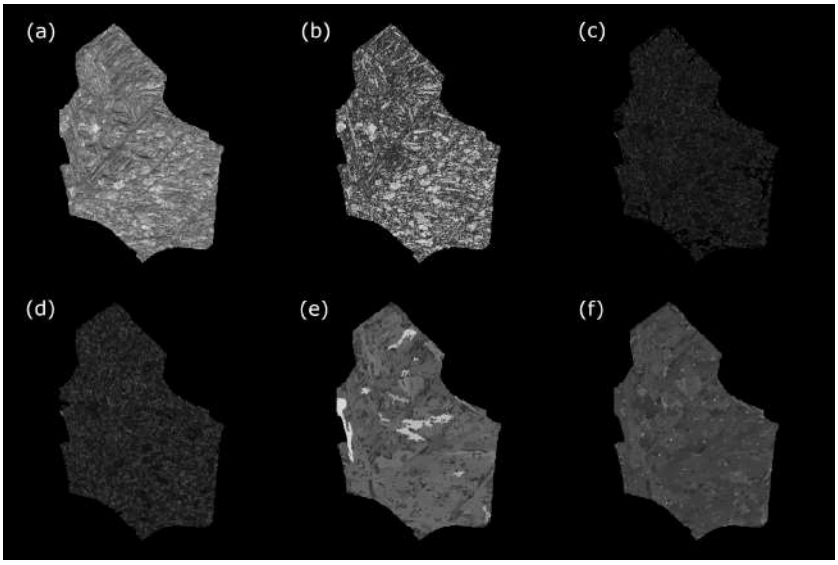

Figure 3.6: EBSD maps of object number 49 in sample C01, showing: (a) IQ, (b) CI, (c) KAM1, (d) KAM3, (e) GOS, (f) GAM.

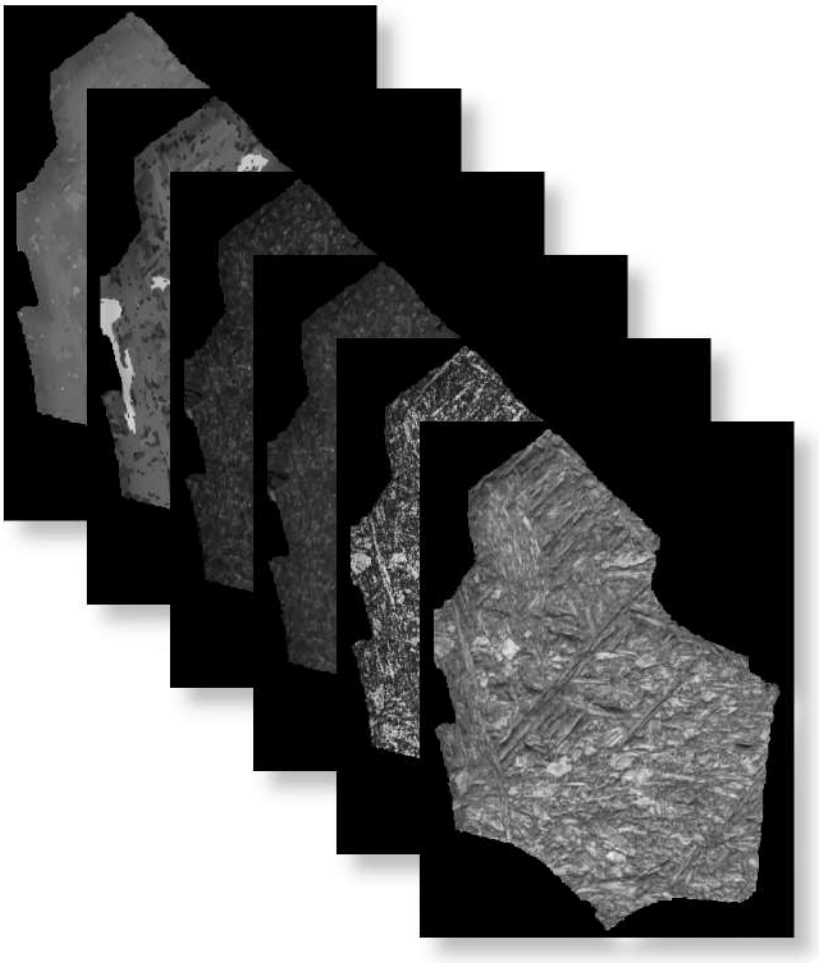

Figure 3.7: Visualization of the EBSD maps matrix for object 49 in sample C01, showing (from front to back): IQ, CI, KAM1, KAM3, GOS and GAM.

A graphical user interface (GUI) was developed to enable the quick creation of datasets from registered images. The GUI allows for the extraction of LOM, SEM, and EBSD objects, and also includes several additional functionalities. These functionalities include the ability to visualize the extracted objects in a unified interface, the capability to train basic RF or SVM models locally for the classification of objects such as bainite or martensite, and the ability to apply these models to predict the label of new objects.

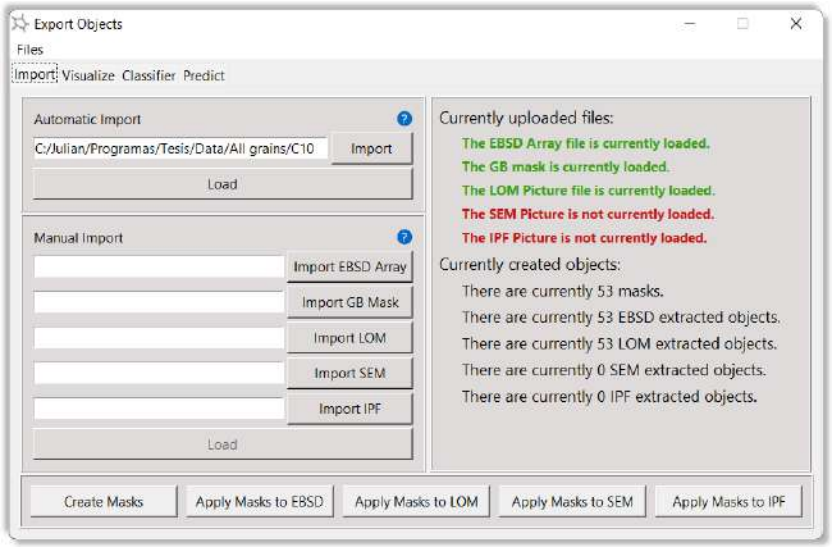

Figure 3.8: GUI for importing the registered images and extracting the objects.

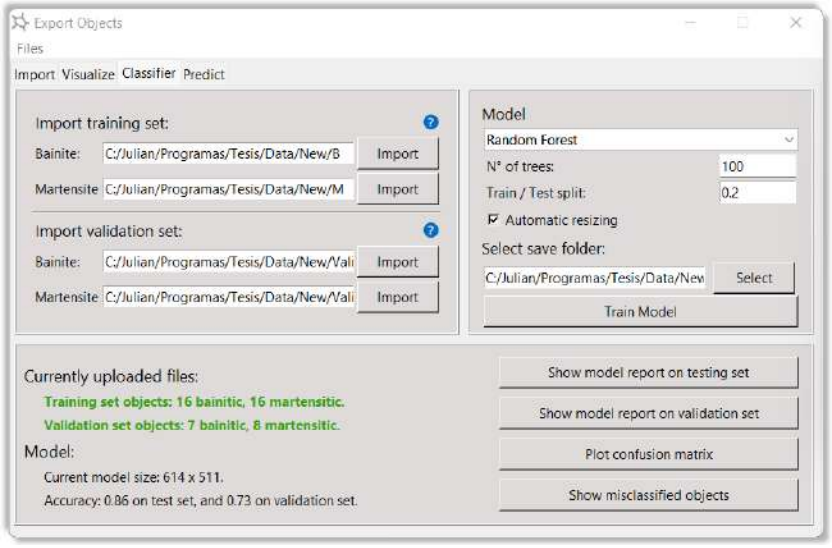

Figure 3.9: GUI for training RF and SVM models on selected datasets.

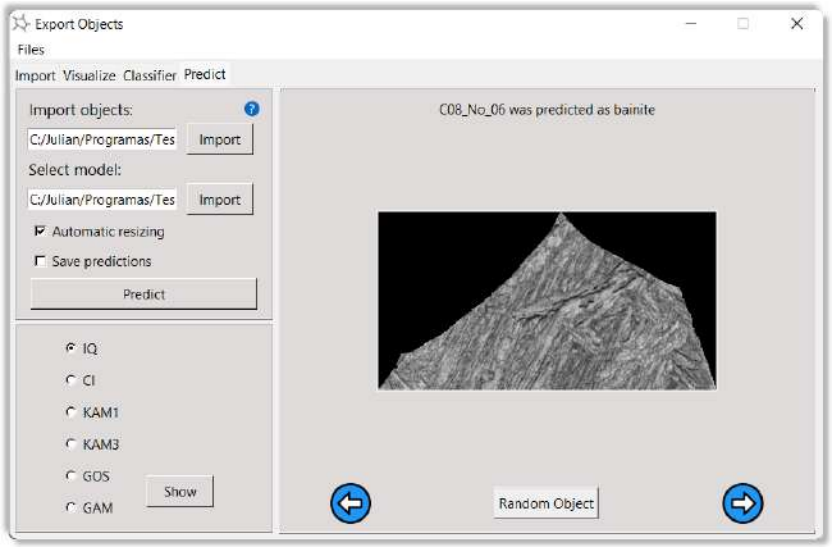

Figure 3.10: GUI for predicting the class of new objects with a selected model.

The final datasets consist of matrices that include the 6 EBSD maps of objects from specimens A and C. For specimens A, the objects from A1 and A2 were classified as martensite, and a total of 83 objects were identified as such. Meanwhile, objects from A7, A9, and A11 were classified as bainite, with a total of 166 objects identified as such. Therefore, the dataset for specimen A consists of 249 objects.

For specimens of composition C, samples C1, C3, and C5 were identified as martensite, with a total of 224 objects. In contrast, specimens C8 and C10 were classified as bainite, with a total of 104 objects. Thus, the dataset for specimen C comprises a total of 328 objects.

3.3.5 Ground-truth

A series of data acquired from the samples, images, and EBSD maps for each object was submitted to a group of experts in order to obtain labels for every object.

Firstly, the TTT diagram for the given composition of the samples and the cooling curve for each sample was provided. Figures 3.11 and 3.12 show the TTT diagrams for samples A and C, respectively. Additionally, dilatometry test

results for each sample were provided. Figure 3.13 and 3.14 show two examples of these curves.

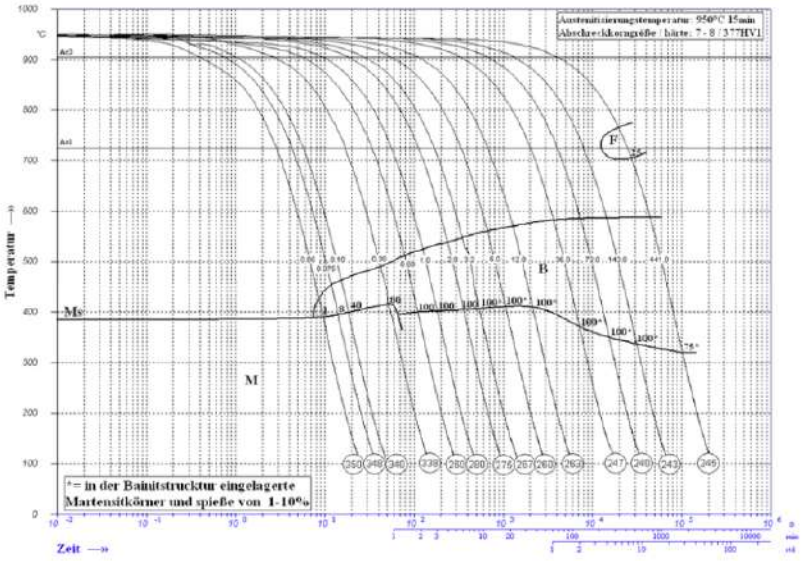

Figure 3.11: TTT diagram with the cooling curves for every A sample.

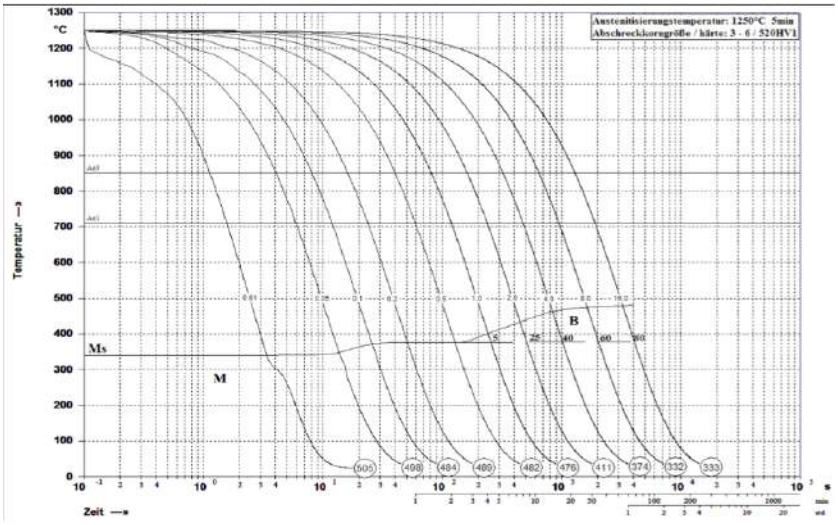

Figure 3.12: TTT diagram with the cooling curves for every C sample.

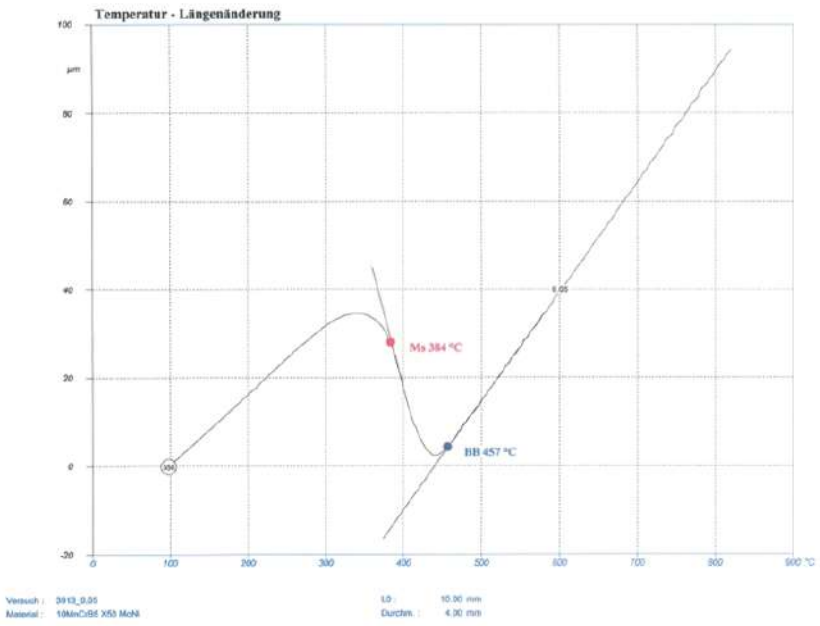

Figure 3.13: Dilatometry curve for sample A1.

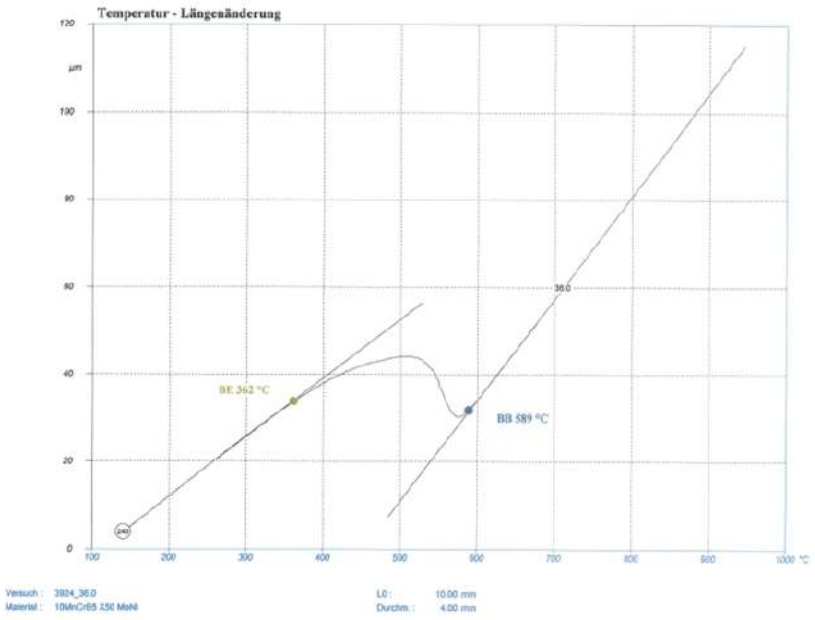

Figure 3.14: Dilatometry curve for sample A11.

Finally, LOM and SEM images, and EBSD maps were provided for each object, along with misorientation angles distributions for each grain. Figures 3.15 and 3.16 show this data for object 24 of sample A1 and object 47 of sample A11, respectively. Furthermore, Figures 3.17 and 3.18 show this data for object 19 of sample C1 and object 46 of sample C10, respectively.

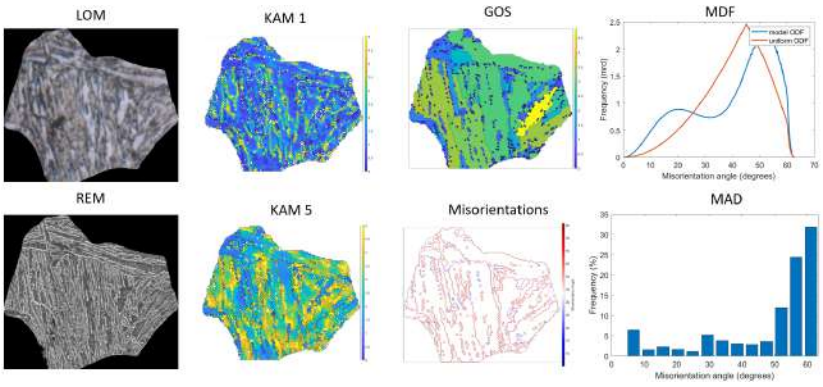

Figure 3.15: LOM and SEM images, EBSD maps, and misorientation angle distribution for object 24 of sample A1, identified as martensite.

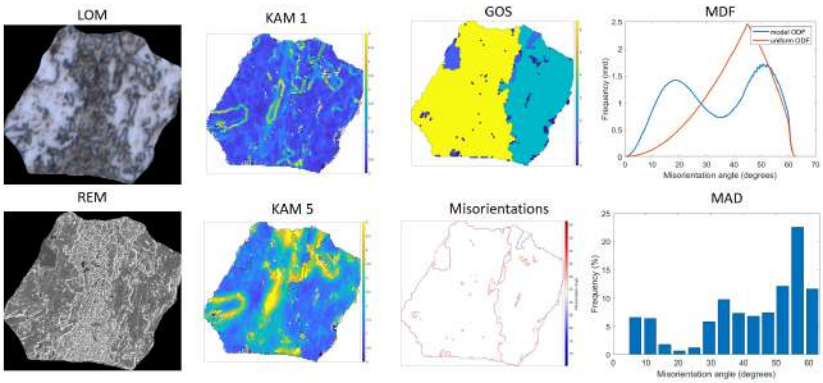

Figure 3.16: LOM and SEM images, EBSD maps, and misorientation angle distribution for object 47 of sample A11, identified as bainite.

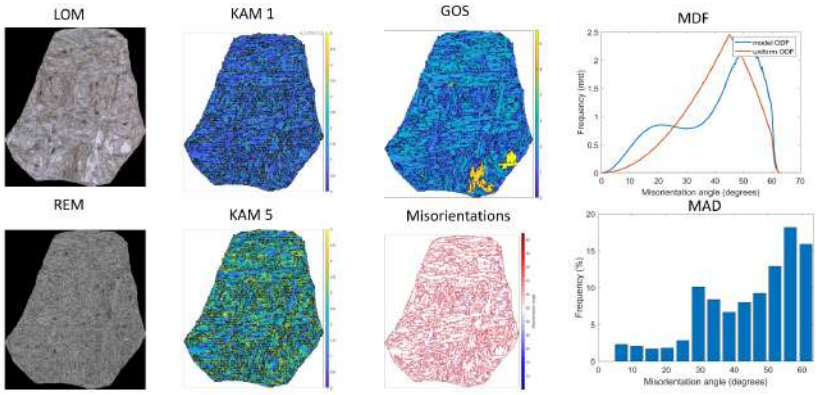

Figure 3.17: LOM and SEM images, EBSD maps, and misorientation angle distribution for object 19 of sample C1, identified as martensite.

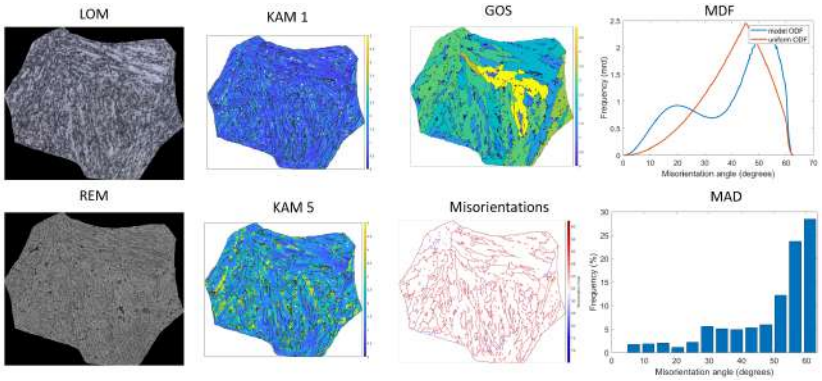

Figure 3.18: LOM and SEM images, EBSD maps, and misorientation angle distribution for object 46 of sample C10, identified as bainite.

With all this information, an objective classification of the objects became possible. The results of the dilatometry test, along with the cooling path indicated by the TTT diagram for each composition, provided valuable insights for determining the class of each object. Additionally, through visual inspection, the presence of a lath-like structure characteristic of martensite in certain objects, such as the one shown in Figure 3.15, could be identified. Moreover, KAM measurements revealed higher misorientation in martensitic objects compared to bainite, and the misorientation angle distribution exhibited distinct patterns.

Chapter 4

Results

4.1 Support Vector Machine

Two Support Vector Machine classifiers were trained separately for specimens A and C using the SVC function of the Scikit-learn Python package [63]. The input for this model is a one-dimensional vector with n features. To obtain this vector from the stack of EBSD maps, all objects were first resized to have the same dimensions as the largest object in terms of height and width. This was achieved by padding the necessary amount of zeros. Once all objects had the same dimensions, the 3-dimensional matrix was flattened into a one-dimensional vector.

The SVM classifier for samples A was trained using 249 objects, consisting of 166 bainite and 83 martensite samples. The largest A object had a height of 333 pixels, while the widest had a width of 375 pixels. Therefore, all objects were resized to a uniform size of $333 \times 375 \times 6$. Consequently, the flattened vector comprised 749,250 features.

The model was trained and tested using k-fold cross-validation with k=5 folds. The best results were found using the linear kernel. The summary of results is presented in Figure 4.1. For each fold, the training accuracy was 1, while the testing accuracy was 0.94, 1, 1, 0.94, and 0.89, respectively, resulting in an average testing accuracy of 0.95 with a standard deviation of 0.04.

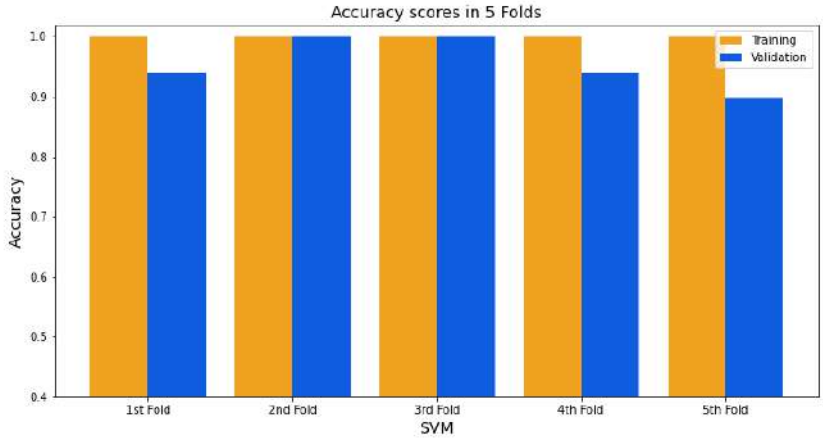

Figure 4.1: Bar plot of the accuracy of training and validation sets for each fold of the SVM classifier of samples A.

The SVM classifier for samples C was trained using 328 objects, consisting of 104 bainite and 224 martensite samples. The largest C object had a height of 614 pixels, while the widest had a width of 683 pixels. Therefore, all objects were resized to a uniform size of $614 \times 683 \times 6$. Consequently, the flattened vector comprised 2,516,172 features.

The model was trained and tested using k-fold cross-validation with k=5 folds. The best results were found using the linear kernel. The summary of results is presented in Figure 4.2. For each fold, the training accuracy was 1, while the testing accuracy was 0.95, 0.90, 0.89, 0.93, and 0.92, respectively, resulting in an average testing accuracy of 0.92 with a standard deviation of 0.03.

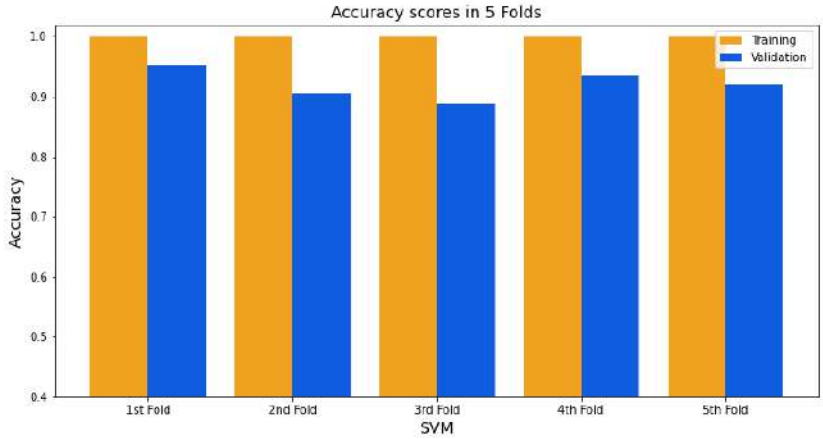

Figure 4.2: Bar plot of the accuracy of training and validation sets for each fold of the SVM classifier of samples C.

Most of the misclassified objects in the C group were small bainite objects or those located at the edges of the full image. These objects were erroneously labeled as martensite. Excluding objects that are cut by the micrograph's boundary is reasonable as these grains do not provide a complete representation of the grain structure. However, the accuracy of the models improves as the number of training samples increases, and the ratio of cut grains to whole grains is relatively low. Furthermore, most incomplete objects were correctly classified. Therefore, it was concluded that it is better to retain these objects in both the training and test sets.

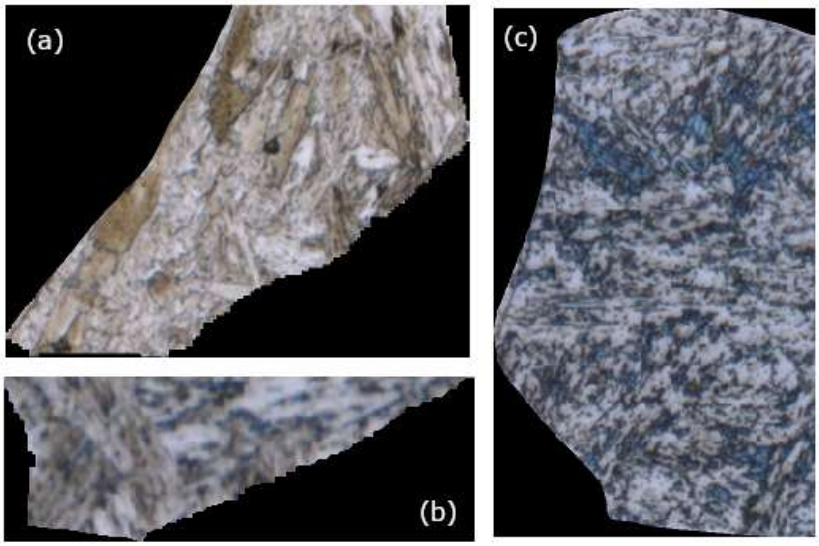

Figure 4.3: (a) LOM image showing object 62 of the C01 sample labeled as martensite but misclassified as bainite. (b) LOM image displaying object 49 of the C08 sample labeled as bainite but misclassified as martensite. (c) LOM image showing object 23 of the C10 sample labeled as bainite but mistakenly classified as martensite.

In summary, both SVM models achieved high accuracy despite not taking into account the spatial nature of the data. The only hyperparameter that needs to be defined is the kernel type, and it was found that the linear kernel yielded better accuracy. Moreover, SVM models are relatively easy to implement and train, and they offer fast and efficient classification. The somewhat lower efficiency observed for the C-samples classifier can be attributed to the higher dimensionality of the input vectors due to the larger size of the matrices, which can make the classification task more challenging.

4.2 Random Forest

Two Random Forest classifiers were trained separately for specimens A and C using the Scikit-learn Python package's RandomForestClassifier function [63]. Like the SVM classifiers, these models also require a one-dimensional vector with n features as input. Therefore, the same method used for obtaining the

SVM classifier input vector was employed. Both RF classifiers were trained using the same dataset as the SVM models, with the flattened vectors having the same number of objects and features.

In the case of RF models, an important hyperparameter, the number of trees (n) in the forest, had to be defined. To select this parameter, 10 models with random train/test splits were trained for each number of trees, ranging from 10 to 500 in increments of 10. The mean accuracy and standard deviations were recorded and presented in Figure 4.4. It was observed that the accuracy of the RF models increases with the number of trees until it reaches a maximum value asymptotically. After 100 trees, the accuracy oscillates around a certain value. The optimal number of trees was determined to be $n = 110$, which yielded high accuracy and low variance across random train/test splits for both sets of samples.

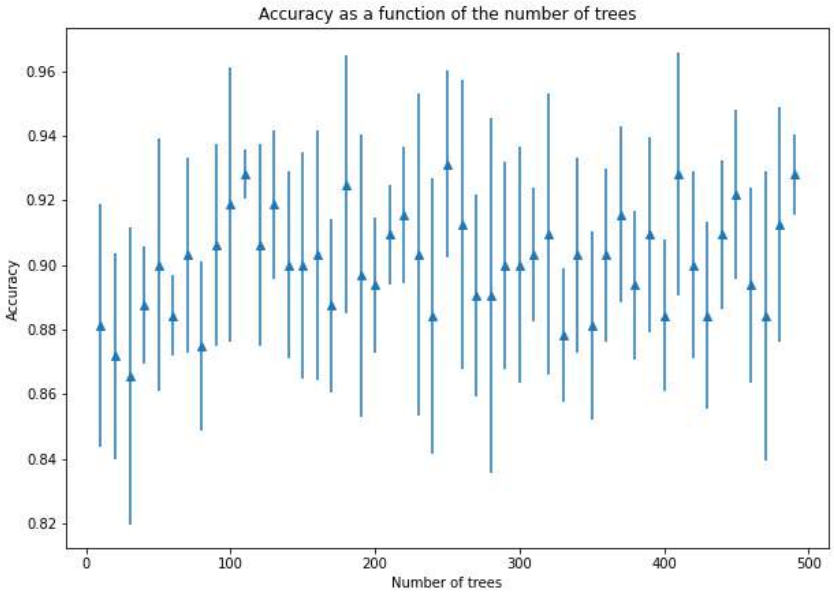

Figure 4.4: Accuracy and standard deviation of RF forest models of samples C as a function of the number of trees n.

After finding the optimal number of trees, a classifier for the A samples

was trained and tested using k-fold cross-validation with k=5 folds. The summary of results is presented in Figure 4.5. For each fold, the training accuracy was 1, while the testing accuracy was 0.96, 0.94, 1, 0.70 and 0.89, respectively, resulting in an average testing accuracy of 0.89 with a standard deviation of 0.11.

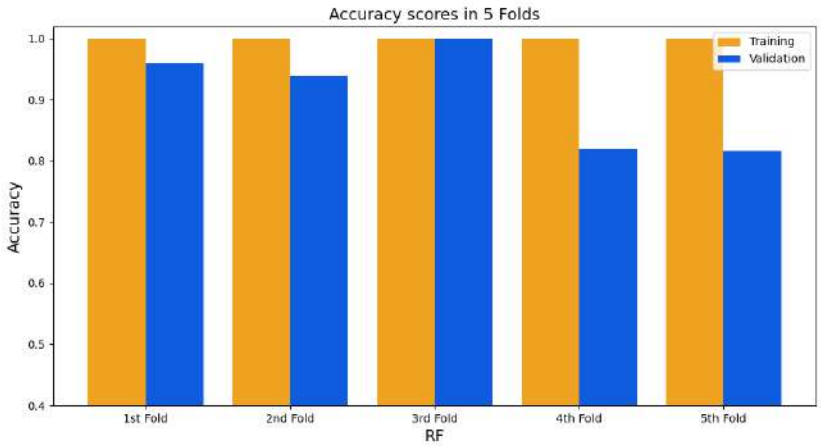

Figure 4.5: Bar plot of the accuracy of training and validation sets for each fold of the RF classifier of samples A.

The classifier of samples A misclassified only the martensitic objects, which can be attributed to the class imbalance in the dataset. However, no misclassification occurred for objects on the edges of the micrograph. Examples of misclassified objects in this model can be observed in Figure 4.6.

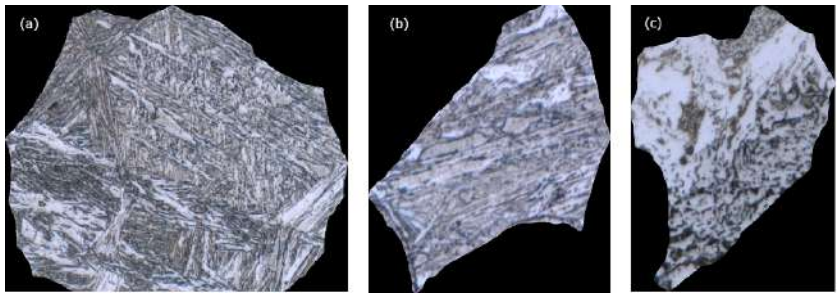

Figure 4.6: (a) LOM image showing object 49 of the A01 sample labeled as martensite but misclassified as bainite. (b) LOM image displaying object 27 of the A02 sample labeled as martensite but misclassified as bainite. (c) LOM image showing object 157 of the A09 sample labeled as bainite but mistakenly classified as martensite.

The same procedure was followed to train the classifier of C samples. The summary of results is presented in Figure 4.7. For each fold, the training accuracy was 1, while the testing accuracy was 0.88, 0.84, 0.86, 0.94 and 0.92, respectively, resulting in an average testing accuracy of 0.88 with a standard deviation of 0.04.

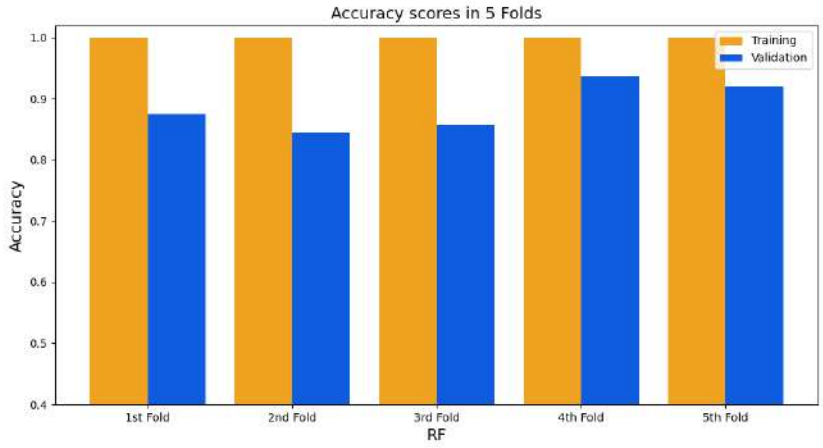

Figure 4.7: Bar plot of the accuracy of training and validation sets for each fold of the RF classifier of samples C.

In this case, all the misclassified objects were from samples C08 and C10, which were objects labeled as bainite. This misclassification is likely due to the class imbalance present in the dataset. Additionally, some objects from the edges of the micrograph were also misclassified. Examples of these misclassified objects in this model can be observed in Figure 4.8.

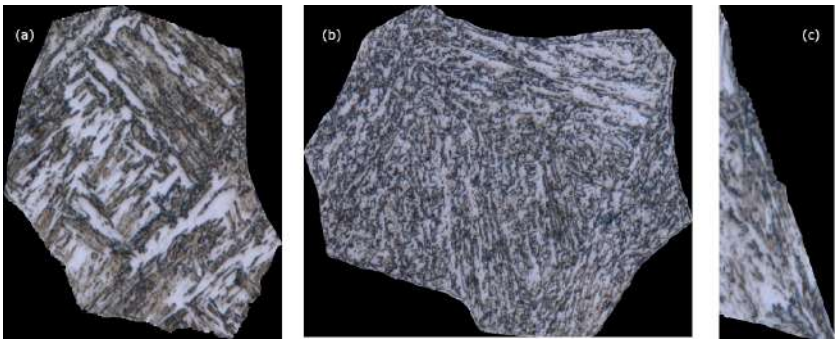

Figure 4.8: Examples of misclassified objects in the Random Forest classifier of samples C. (a) LOM image of object 39 from the C08 sample labeled as martensite but classified as bainite. (b) LOM image of object 46 from the C10 sample labeled as martensite but classified as bainite. (c) LOM image of object 32 from the C10 sample labeled as bainite but classified as martensite.

In summary, although the RF models had good accuracy, they performed worse than the SVM models. The training process for RF models took slightly longer than for SVM models because the optimal number of trees had to be found. The RF models were also found to be more susceptible to class imbalance than SVM models. Moreover, while both RF and SVM models are easy to implement and have high accuracy, they are not specifically designed to work with images, and they may struggle to find patterns in more complex data. Therefore, their accuracy may decrease when classifying multiple classes, such as different types of bainite or ferrite.

4.3 Deep Neural Network

To improve accuracy and enable multiclass classification, training a deep neural network was considered. DNNs are theoretically better equipped to recognize

complex patterns in data than RF and SVM models. However, they are not specifically designed for image processing.

Despite testing several DNN architectures, none achieved high accuracies. Furthermore, the training processes did not exhibit signs of improvement.

The primary issue with using a DNN for image processing is that flattened matrices lead to a high number of input features, resulting in many parameters even in the first layer. Consequently, these architectures had millions of parameters, which necessitate a vast number of training examples. This problem can be mitigated by employing a convolutional neural network.

4.4 Convolutional Neural Network

Given the complexity and high dimensionality of the data, convolutional neural networks are a natural choice for this problem. CNNs are particularly adept at identifying and extracting relevant different level features from images, making them an ideal tool for this classification tasks. Moreover, CNNs can significantly reduce the number of parameters required for the model, enabling effective training with a limited number of examples. However, training a CNN is a computationally expensive process that requires a deep understanding of machine learning fundamentals, including defining an appropriate architecture and selecting hyperparameters that optimize performance.

To optimize the performance of the CNN models, various hyperparameters were experimented with, including the number of convolutional layers, the number of filters in each layer, the kernel size, and the pooling method. The general strategy to improve the performance of the model was to reduce the number of parameters as much as possible while maintaining a complex enough model to learn the features of the data. The hyperparameters were tuned based on their effect on the performance metrics of the model. Specifically, the aim was to maximize accuracy while minimizing overfitting and computational complexity. The hyperparameters that resulted in the best performance were selected for the final model.

The same CNN architecture is used to classify both specimens A and C, with the only difference being the input size. The architecture starts with two convolutional layers, each with 16 filters and activation layers. This is followed

by a max pooling layer with a sliding window of size 2×2, which reduces the input's height and width by half. Then, two more convolutional layers with 32 filters and activation layers follow, and another max pooling layer with the same sliding window size. After that, two more convolutional layers with 64 filters and activation layers follow, and then another max pooling layer with the same sliding window size. The features are then further reduced using average pooling with a kernel size of 5×5, and the resulting matrix is flattened. The final layers consist of a fully connected network with 64, 32, and 16 neurons, and a softmax layer with two outputs to predict the class. A visual scheme of the CNN architecture can be found in Figure 4.9.

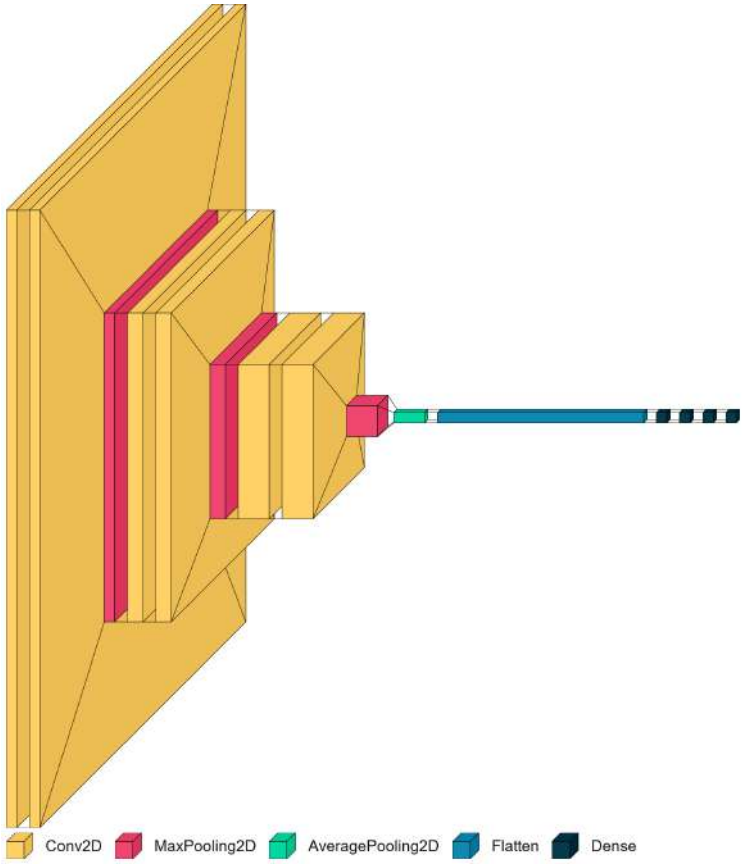

Figure 4.9: Visualization of the CNN architecture for both models created using Python's visualkeras module [64]. Feature shapes are not to scale for visual clarity.

Both models use a kernel size of 3×3 in every convolution layer. The ReLU activation function is applied to every layer, except for the final classification layer, which uses a softmax function. The "padding" argument in every convolution is set to "same" so that downsampling occurs only in the pooling layers and not in the convolutional layers. The Adam optimizer is used with a learning rate of 0.000025, and the sparse categorical cross-entropy loss function is used for training. The batch size was set to 1.

All CNNs were trained using a Google Colab Pro account, which provided access to a Tesla T4 GPU and 25 GB of RAM. This setup enabled faster training compared to a local environment.

The classifier for samples A was trained using data augmentation, which involved rotating some objects by random angles to increase the size of the training set. The input shape was $333 \times 375 \times 6$, and the dataset consisted of 498 objects. With this input shape and the architecture mentioned above, the total number of parameters in the model was 112,082. The model was trained for 30 epochs and tested using k-fold cross-validation with 5 folds. The testing and training accuracies are shown in Figure 4.10. The testing accuracy for each fold can also be found on Table 4.1, where they can be compared with the results for the other models. The mean testing accuracy was 0.990, with a standard deviation of 0.007.

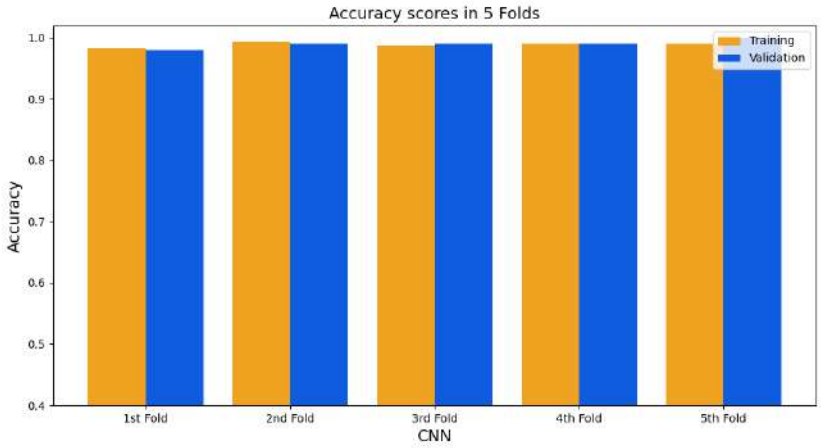

Figure 4.10: Bar plot of the accuracy of training and validation sets for each fold of the CNN classifier of samples A.

The accuracy plot for the second fold as a function of training epochs is shown in Figure 4.11, while the accuracy plots for the other folds can be found in the appendices (Figures 5.1, 5.2, 5.3, and 5.4). The loss plot for the second fold is shown in Figure 4.12, and the remaining loss plots can be found in the appendices (Figures 5.5, 5.6, 5.7, and 5.8).

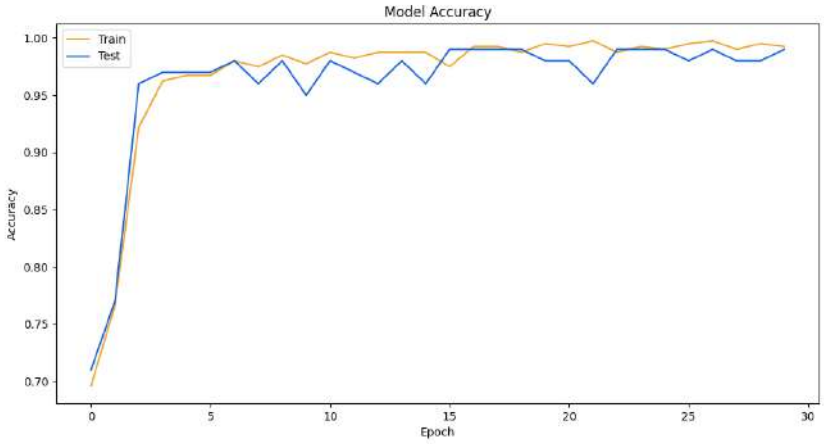

Figure 4.11: Accuracy plot for the second fold as a function of training epochs for samples A.

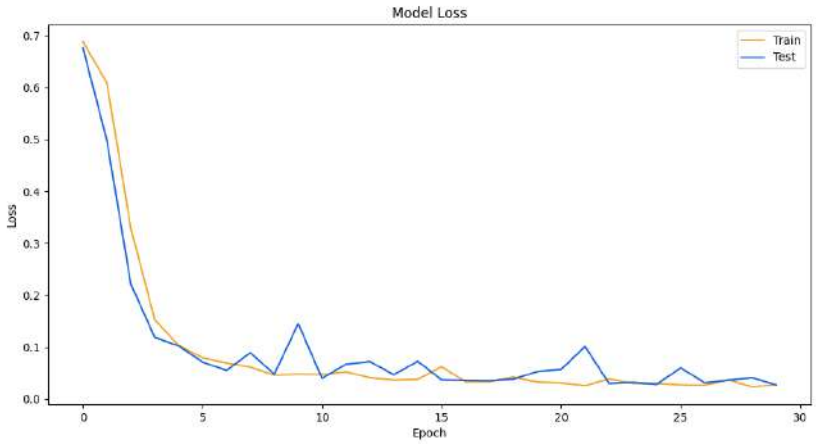

Figure 4.12: Loss plot for the second fold as a function of training epochs for samples A.

This model achieved very high accuracy. However, it was observed that the train and test accuracy fluctuated due to the limited number of training objects in the dataset. To address this issue, data augmentation was considered, but it was limited by the available RAM. Therefore, the model's performance and reproducibility can be further improved with more data. Another alternative approach to utilize more data augmentation could involve resizing the objects to smaller dimensions, but this may result in information loss and would require revalidating the model's robustness.

The input shape of classifier for samples C was $700 \times 700 \times 6$ and the dataset consisted of 316 objects. The larger size of the objects prevented the use of data augmentation as it required a large amount of RAM. With this input shape and the architecture used, the total number of parameters in the model was 275,922. The model was trained for 50 epochs and tested using k-fold cross-validation with 5 folds. The testing and training accuracies are shown in Figure 4.13. The testing accuracy for each fold can also be found on Table 4.1, where they can be compared with the results for the other models. The mean testing accuracy was 0.991, with a standard deviation of 0.009.

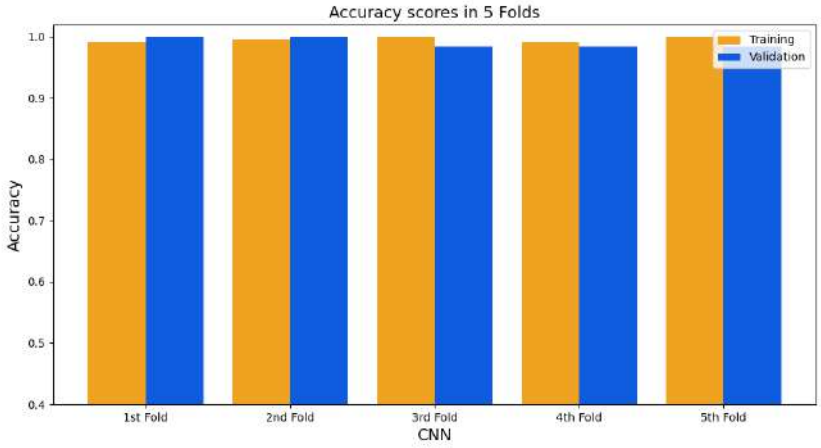

Figure 4.13: Bar plot of the accuracy of training and validation sets for each fold of the CNN classifier of samples C.

Figure 4.14 shows the accuracy plot for the first fold as a function of training epochs. The accuracy graphs for the other folds can be found in the appendices in Figures 5.9, 5.10, 5.11, and 5.12. The loss plot for the first fold is shown in Figure 4.15, and the remaining loss plots can be found in the appendices in Figures 5.13, 5.14, 5.15 and 5.16.

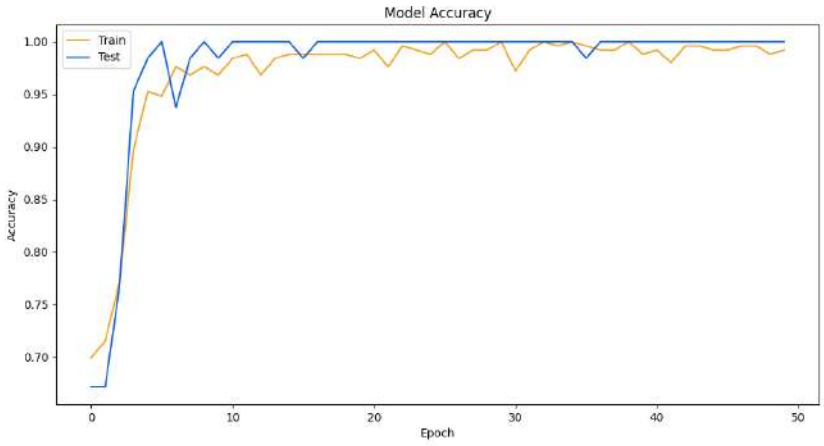

Figure 4.14: Accuracy plot for the first fold as a function of training epochs for samples C.

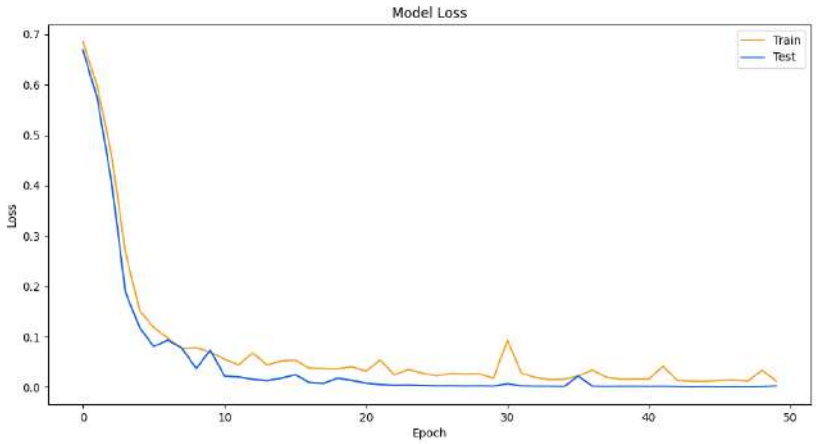

Figure 4.15: Loss plot for the first fold as a function of training epochs for samples C.

This model also achieved very high accuracy but showed more fluctuations in accuracy and loss due to the smaller number of objects. Therefore, the model's performance and reproducibility can also be further improved with

more data.

Figures 4.16 and 4.17 depict two inputs used in the CNN models. Figure 4.16 shows object 49 from sample C01, labeled as martensite, while Figure 4.17 shows object 20 from sample C10, labeled as bainite. Some visible differences can be observed between the two figures, such as the IQ and CI maps being brighter for the object labeled as bainite. This is because bainite typically has a lower dislocation density than martensite, resulting in higher quality diffraction patterns. Figures 4.18 and 4.19 show a group of selected feature maps for these to objects. As this is the output of the first convolution layer, the feature maps do not differ much from the inputs and there are no marked differences between the objects

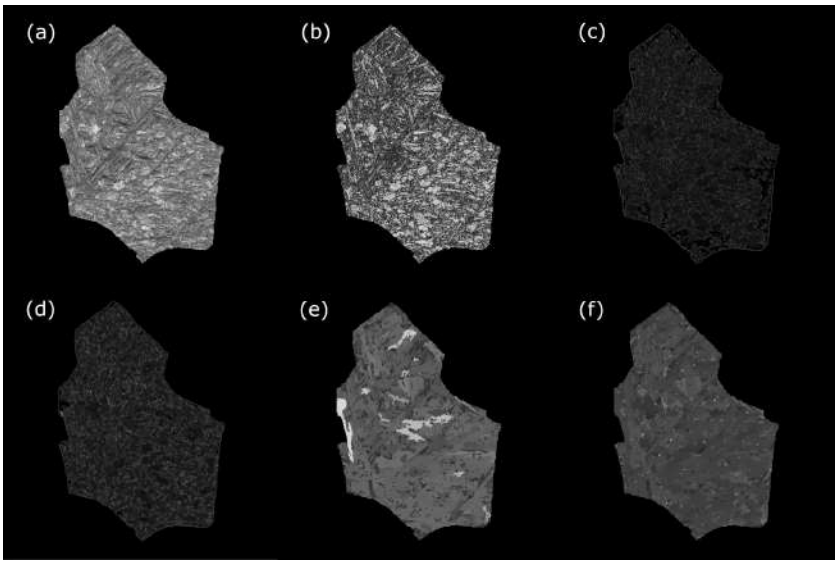

Figure 4.16: EBSD maps of object number 49 in sample C01, showing: (a) IQ, (b) CI, (c) KAM1, (d) KAM3, (e) GOS, (f) GAM.

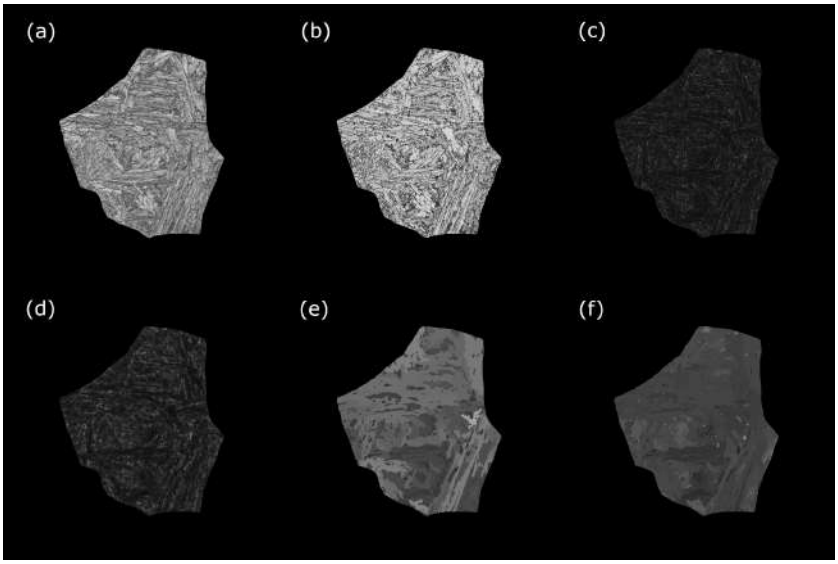

Figure 4.17: EBSD maps of object number 20 in sample C10, showing: (a) IQ, (b) CI, (c) KAM1, (d) KAM3, (e) GOS, (f) GAM.

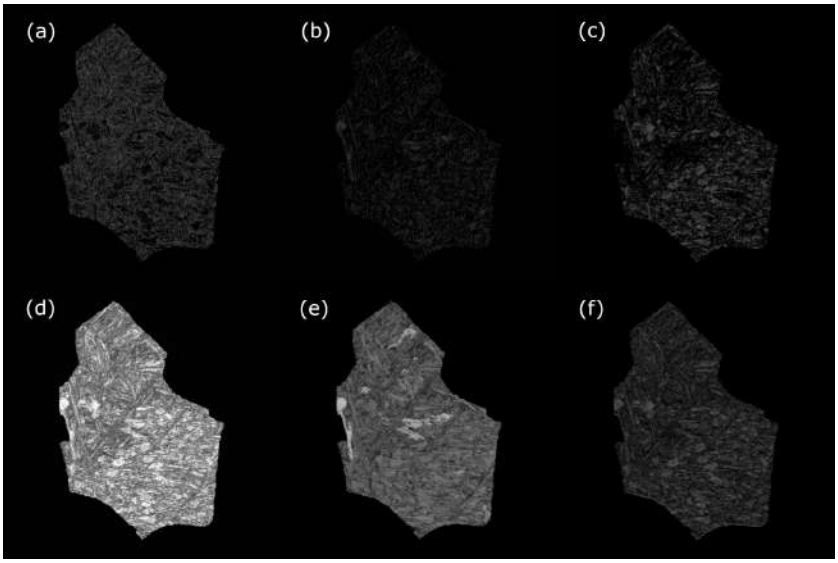

Figure 4.18: Feature maps number (a) 1, (b) 5, (c) 6, (d) 7, (e) 8, and (f) 13 obtained after the first convolution layer for object 49 of sample C01.

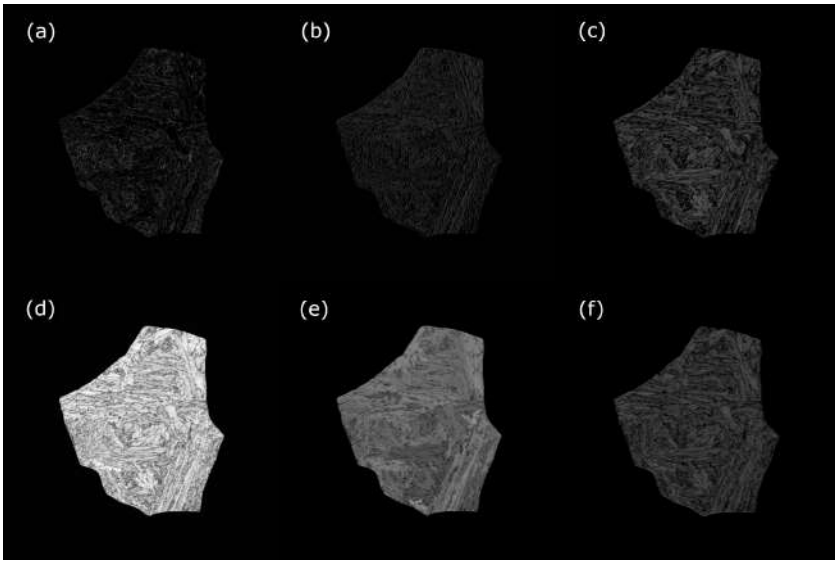

Figure 4.19: Feature maps number (a) 1, (b) 5, (c) 6, (d) 7, (e) 8, and (f) 13 obtained after the first convolution layer for object 20 of sample C10.

Furthermore, Figures 4.20 and 4.21 display a selected group of feature maps obtained after the third convolution for object 49 of sample C01 and object 20 of sample C10, respectively. A greater difference between the objects can be observed here since the feature maps correspond to stages closer to the classification stage. The feature maps of the object labeled as bainite tend to be brighter, while some dark marked features within the feature maps of the object labeled as martensite appear lighter in the object labeled as bainite and vice versa.

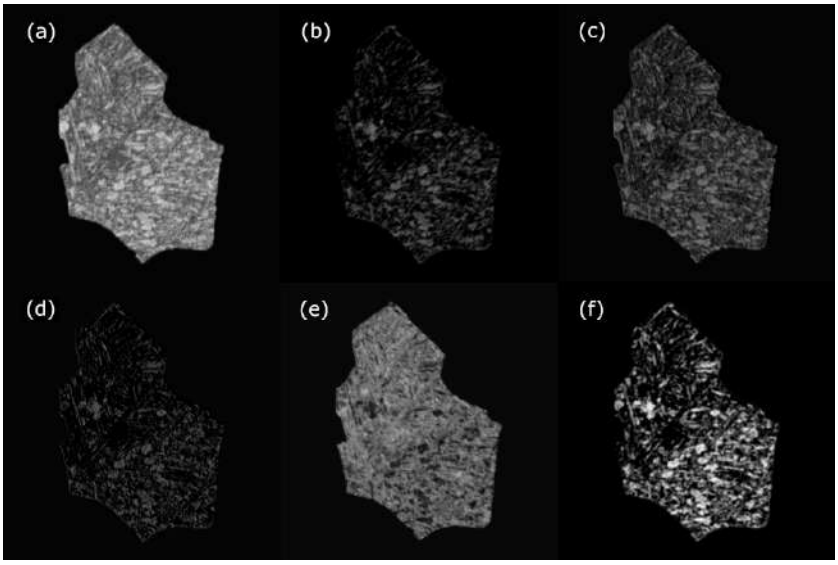

Figure 4.20: Feature maps number (a) 1, (b) 4, (c) 5, (d) 12, (e) 17, and (f) 27 obtained after the third convolution layer for object 49 of sample C01.

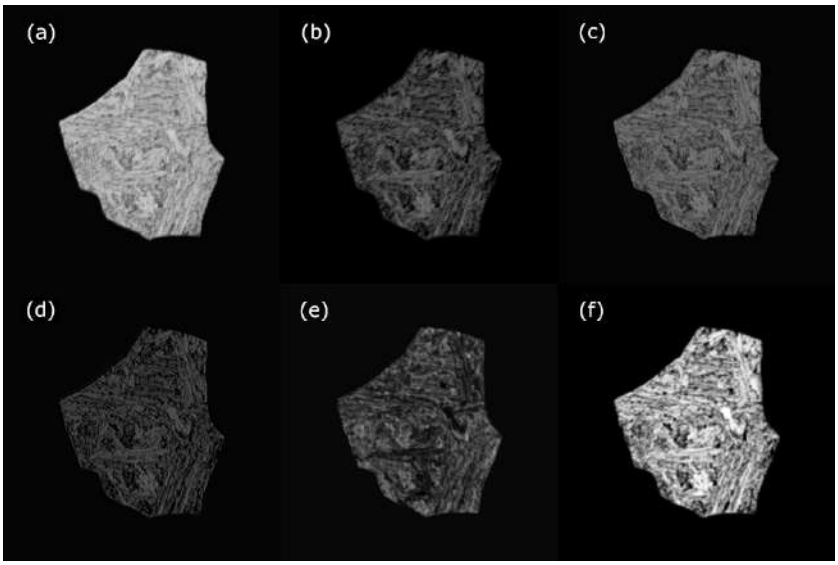

Figure 4.21: Feature maps number (a) 1, (b) 4, (c) 5, (d) 12, (e) 17, and (f) 27 obtained after the third convolution layer for object 20 of sample C10.

Finally, a selected group of feature maps from the aforementioned objects after the sixth and last convolution layer are shown in Figures 4.22 and 4.23. This layer is immediately followed by the final pooling and flattening of the matrix, which occurs prior to the fully connected layers and the classification stage. A marked difference can be observed between the feature maps of the two objects. The objects that appear almost completely white in the object labeled as martensite look completely dark in the corresponding feature map for the object labeled as bainite. Similarly, features within objects that appear dark in one object tend to appear bright in the other. Some feature maps detect edges, while feature map number 30 appears to be showing some kind of texture that looks finer on the object classified as martensite. The differences between these feature maps provide insight into how the final densely connected network can distinguish these two classes.

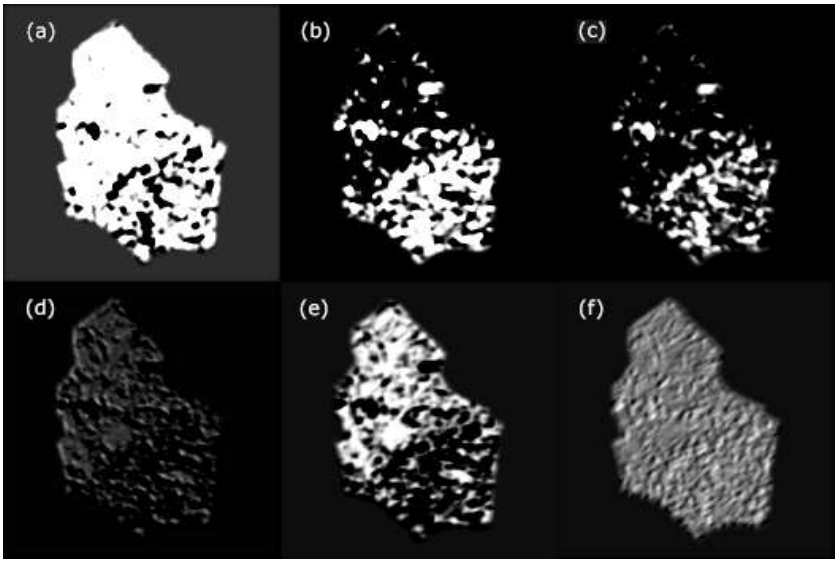

Figure 4.22: Feature maps number (a) 2, (b) 4, (c) 6, (d) 7, (e) 20, and (f) 30 obtained after the sixth and last convolution layer for object 49 of sample C01.

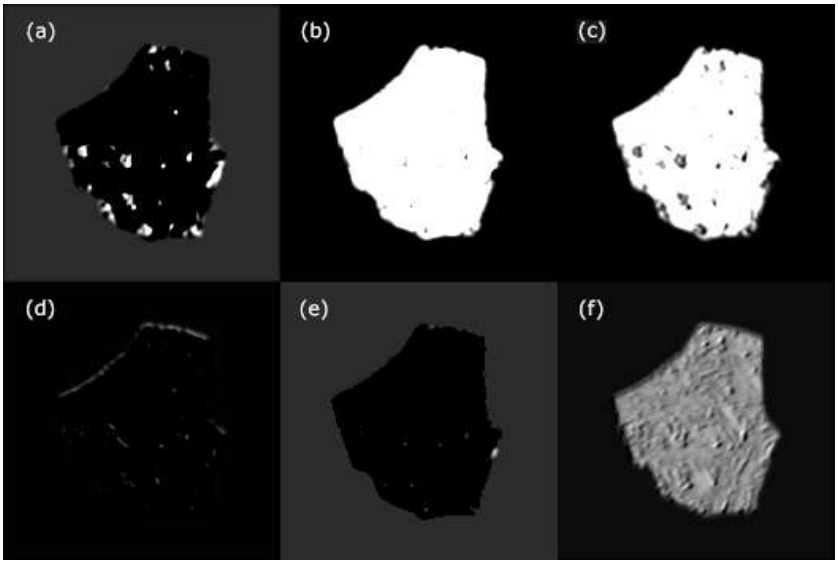

Figure 4.23: Feature maps number (a) 2, (b) 4, (c) 6, (d) 7, (e) 20, and (f) 30 obtained after the sixth and last convolution layer for object 20 of sample C10.

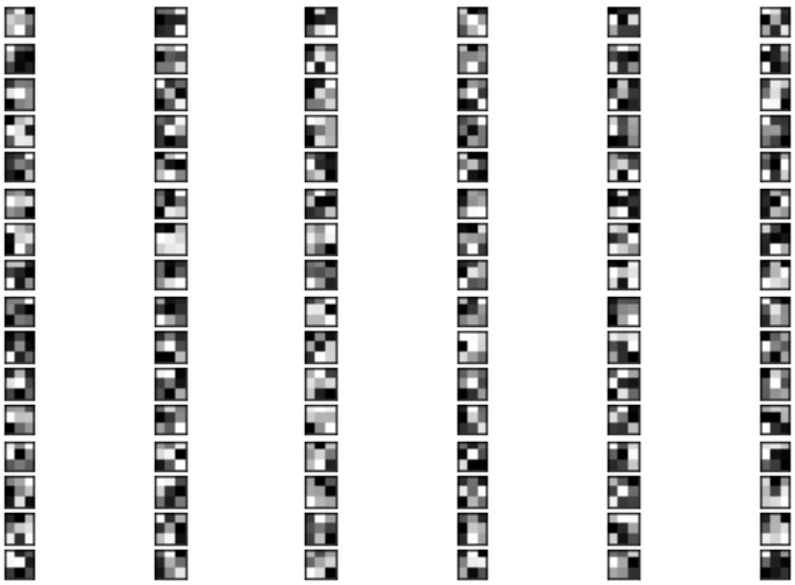

Figure 4.24: Illustration of the 16 filters in the first convolutional layer, each with a shape of $3 \times 3 \times 6$.

The high efficiency achieved by this architecture opens up possibilities for its application in more complex tasks and industrial environments. Firstly, the promising performance of the model suggests its potential for generalizing to classification tasks involving multiple classes. A model capable of objectively and reproducibly classifying various steel phases, such as different types of ferrite, different types of bainite, pearlite, and martensite, holds significant scientific and industrial value.

Secondly, the versatility of this architecture allows for adaptation to different input sources, which can lead to performance improvements or enable the use of more rapidly obtainable data. For instance, these CNNs can be trained using each EBSD map, providing insight into the crucial parameters for phase classification and potentially enhancing performance. Furthermore, employing the same architecture with LOM images, which require less time to acquire, would increase its feasibility in industrial-scale applications.

In conclusion, the architecture performed very well on the testing sets, achieving higher efficiency than the SVM and RF models. While the RF and SVM models may be easier to implement, this architecture shows promise for more complex tasks, such as multi-class classification. Furthermore, the efficiency and consistency of results can be improved by increasing the amount of training and testing data.

Samples	Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg. Test Acc.	Dev. Test Acc.
A	SVM	0,94	1,00	1,00	0,94	0,90	0,9556	0,0440
	RF	0,96	0,94	1,00	0,71	0,89	0,8990	0,1151
	CNN	0,98	0,99	0,99	0,99	1,00	0,9900	0,0071
C	SVM	0,95	0,91	0,90	0,92	0,94	0,9243	0,0206
	RF	0,88	0,84	0,86	0,94	0,92	0,8866	0,0403
	CNN	1,00	1,00	0,98	0,98	0,98	0,9905	0,0087

Table 4.1: Accuracy on testing set for every fold of every model.

Chapter 5

Conclusion

In this study, two groups of samples with distinct compositions were effectively investigated and characterized. Each composition group consisted of five samples, each with varying cooling rates resulting in different microstructures. A ROI was carefully selected for each sample, and images were acquired using both LOM and SEM. Additionally, EBSD maps were obtained for each sample, providing valuable insights into its crystallographic orientation.

To overcome the inherent challenges associated with conventional microstructural characterization, such as subjectivity and lack of reproducibility, a modern correlative approach was employed. This approach involved overlaying LOM and SEM images with EBSD maps of the designated ROIs, enabling a comprehensive and integrated analysis. By integrating diverse sources of information based on distinct physical principles, a deeper understanding of the sample's microstructure was achieved, facilitating a more objective determination of ground truths. Additionally, this approach facilitated the creation of a grain boundary mask, allowing for the extraction of individual bainite or martensite objects.

Python scripts and GUIs were developed for the efficient acquisition and manipulation of datasets. These GUIs enable the rapid extraction of objects from the various methods. Additional features were also incorporated to make the software suitable for both dataset creation and local model training and application.

Multiple machine learning models were trained for both sets of samples. The process of model selection and hyperparameter optimization required a comprehensive understanding of various machine learning techniques.

Initially, RF and SVM models were trained due to their ease of implementation using available Python modules. While these methods are not specifically tailored for this task, they provided a practical option for swift and efficient implementation. The models exhibited satisfactory efficiency, albeit not exceptionally high. It should be noted that their performance may decrease for more complex tasks, such as multi-class classification.

Subsequently, a search was conducted to identify a CNN architecture that could effectively address the task at hand. Through experimentation and tuning of various architectures and hyperparameters, a configuration was determined that showed promising results. This architecture was employed for both sets of samples, with only the input shape differing. The models achieved remarkably high accuracy and reproducibility. Although these models can already be utilized for the objective and reproducible classification of bainite and martensite, it is worth mentioning that further improvements can be made. For instance, incorporating a larger dataset or performing feature selection could enhance their performance.

Furthermore, this architecture demonstrates promising potential for tackling more complex tasks of greater significance in scientific and industrial applications. Particularly, it exhibits promise in the classification of multiple steel phases, including different types of ferrite, bainite, martensite, and pearlite, which will be explored in future research.

References

[1] AJ de DeArdo et al. "New method of characterizing and quantifying complex microstructures in steels". In: *Materials and Manufacturing Processes* 25.1-3 (2010), pp. 33–40.

[2] Sachin L Shrestha et al. "An automated method of quantifying ferrite microstructures using electron backscatter diffraction (EBSD) data". In: *Ultramicroscopy* 137 (2014), pp. 40–47.

[3] D Britz et al. "Identifying and quantifying microstructures in low-alloyed steels: a correlative approach". In: *Metallurgia Italiana* 3 (2017), pp. 5–10.

[4] Jessica Gola et al. "Objective microstructure classification by support vector machine (SVM) using a combination of morphological parameters and textural features for low carbon steels". In: *Computational Materials Science* 160 (2019), pp. 186–196.

[5] Martin Muller, Dominik Britz, and Frank Mucklich. "Machine Learning For Microstructure Classification: How To Assign The Ground Truth In The Most Objective Way." In: *Advanced Materials & Processes* 179.1 (2021), pp. 16–22.

[6] Seyed Majid Azimi et al. "Advanced steel microstructural classification by deep learning methods". In: *Scientific reports* 8.1 (2018), p. 2128.

[7] Johannes Schindelin et al. "Fiji: an open-source platform for biological-image analysis". In: *Nature methods* 9.7 (2012), pp. 676–682.

[8] Ignacio Arganda-Carreras et al. "Consistent and elastic registration of histological sections using vector-spline regularization". In: *Computer Vision Approaches to Medical Image Analysis: Second International ECCV Workshop, CVAMIA 2006 Graz, Austria, May 12, 2006 Revised Papers 2*. Springer. 2006, pp. 85–95.

[9] Björn-Ivo Bachmann et al. "Efficient reconstruction of prior austenite grains in steel from etched light optical micrographs using deep learning and annotations from correlative microscopy". In: *Frontiers in Materials* 9 (2022), p. 1033505.

[10] Harry Bhadeshia and Robert Honeycombe. *Steels: microstructure and properties*. Butterworth-Heinemann, 2017.

[11] D.R. Gaskell. "Allotropy and Polymorphism". In: *Encyclopedia of Condensed Matter Physics*. Ed. by Franco Bassani, Gerald L. Liedl, and Peter Wyder. Oxford: Elsevier, 2005, pp. 8–17.

[12] AG Caesar. *Iron carbon phase diagram*. Feb. 2019. URL: `https://es.wikipedia.org/wiki/Archivo:Iron_carbon_phase_diagram.svg`.

[13] Harshad Kumar Dharamshi Hansraj Bhadeshia. *Bainite in steels: theory and practice*. CRC press, 2019.

[14] Harshad KDH Bhadeshia. *Geometry of crystals, polycrystals, and phase transformations*. CRC press, 2017.

[15] HKDH Bhadeshia. "Martensite and bainite in steels: transformation mechanism & mechanical properties". In: *Le Journal de Physique IV* 7.C5 (1997), pp. C5–367.

[16] Frank Niessen et al. "Parent grain reconstruction from partially or fully transformed microstructures in MTEX". In: *Journal of Applied Crystallography* 55.1 (2022), pp. 180–194.

[17] HKDH Bhadeshia and POSTECH GIFT. *POSCO Lectures: The bainite reaction*. 2010.

[18] LCD Fielding. "The bainite controversy". In: *Materials Science and Technology* 29.4 (2013), pp. 383–399.

[19] ES Davenport and EC Bain. "Transformation of austenite at constant subcritical temperatures". In: *Metallurgical Transactions* 1.12 (1970), pp. 3503–3530.

[20] JM Robertson. "The microstructure of rapidly cooled steel". In: *Journal of the Iron and Steel Institute* 119 (1929), pp. 391–419.

[21] Stanislaw Zajac, Volker Schwinn, and KH Tacke. "Characterisation and quantification of complex bainitic microstructures in high and ultra-high strength linepipe steels". In: *Materials Science Forum*. Vol. 500. Trans Tech Publ. 2005, pp. 387–394.

[22] Günter Petzow. *Metallographic etching: techniques for metallography, ceramography, plastography*. ASM international, 1999.

[23] Bruce L Bramfitt and Arlan O Benscoter. *Metallographer's guide: practice and procedures for irons and steels*. Asm International, 2001.

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

[25] Adam J Schwartz et al. *Electron backscatter diffraction in materials science*. Vol. 2. Springer, 2009.

[26] SI Wright. "Orientation texture". In: (2005).

[27] Stuart I Wright and Matthew M Nowell. "EBSD image quality mapping". In: *Microscopy and microanalysis* 12.1 (2006), pp. 72–84.

[28] Roumen Petrov et al. "Microstructure and texture of a lightly deformed TRIP-assisted steel characterized by means of the EBSD technique". In: *Materials Science and Engineering: A* 447.1-2 (2007), pp. 285–297.

[29] Jinghui Wu et al. "Image quality analysis: A new method of characterizing microstructures". In: *ISIJ international* 45.2 (2005), pp. 254–262.

[30] T Martinez Ostormujof et al. "Deep Learning for automated phase segmentation in EBSD maps. A case study in Dual Phase steel microstructures". In: *Materials Characterization* 184 (2022), p. 111638.

[31] Stuart I Wright, Matthew M Nowell, and David P Field. "A review of strain analysis using electron backscatter diffraction". In: *Microscopy and microanalysis* 17.3 (2011), pp. 316–329.

[32] Yu-Wen Chen et al. "Phase quantification in low carbon Nb-Mo bearing steel by electron backscatter diffraction technique coupled with kernel average misorientation". In: *Materials Characterization* 139 (2018), pp. 49–58.

[33] D Britz et al. "A correlative approach to capture and quantify substructures by means of image registration". In: *Practical Metallography* 54.10 (2017), pp. 685–696.

[34] Barbara Zitova and Jan Flusser. "Image registration methods: a survey". In: *Image and vision computing* 21.11 (2003), pp. 977–1000.

[35] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60 (2004), pp. 91–110.

[36] M Müller, D Britz, and F Mücklich. "Scale-bridging microstructural analysis–a correlative approach to microstructure quantification combining microscopic images and EBSD data". In: *Practical Metallography* 58.7 (2021), pp. 408–426.

[37] Björn Ivo Bachmann. "Prozess-Gefüge-Eigenschaftskorrelationen an ATP Blechen mit modernen Methoden der Gefügeanalyse unter Zuhilfenahme von maschinellem Lernen". MA thesis. Universität des Saarlandes - Luleå Tekniska Universitet, 2021.

[38] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[39] Oliver Theobald. *Machine learning for absolute beginners: a plain English introduction*. Vol. 157. Scatterplot press London, UK, 2017.

[40] Daniel Zhang et al. "The AI index 2021 annual report". In: *arXiv preprint arXiv:2103.06312* (2021).

[41] Jiayi Shen et al. "Artificial intelligence versus clinicians in disease diagnosis: systematic review". In: *JMIR medical informatics* 7.3 (2019), e10010.

[42] Jessica Gola et al. "Advanced microstructure classification by data mining methods". In: *Computational Materials Science* 148 (2018), pp. 324–335.

[43] M Müller et al. *Classification of Bainitic structures using textural parameters and machine learning techniques. Metals 630 (10), 1–19 (2020).*

[44] Kazumasa Tsutsui et al. "Microstructural diagram for steel based on crystallography with machine learning". In: *Computational Materials Science* 159 (2019), pp. 403–411.

[45] S Zaefferer, P Romano, and F Friedel. "EBSD as a tool to identify and quantify bainite and ferrite in low-alloyed Al-TRIP steels". In: *Journal of microscopy* 230.3 (2008), pp. 499–508.

[46] Aurélien Géron. "Hands-on machine learning with scikit-learn and tensorflow: Concepts". In: *Tools, and Techniques to build intelligent systems* (2017).

[47] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[48] Seema Singh. *Understanding the Bias-Variance Tradeoff*. - Towards Data Science. Medium. - Visited on April 3, 2023. URL: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229.

[49] Nello Cristianini, John Shawe-Taylor, et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[50] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[51] Tony Yiu. *Understanding Random Forest*. - Towards Data Science. Medium. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2m.

[52] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[53] Charu C Aggarwal et al. "Neural networks and deep learning". In: *Springer* 10.978 (2018), p. 3.

[54] Tibco. *What is a Neural Network?* - Visited on March 2, 2023. URL: https://www.tibco.com/reference-center/what-is-a-neural-network.

[55] Daphne Cornelisse. *An intuitive guide to Convolutional Neural Networks*. - Free Code Camp - Visited on March 3, 2023. URL: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/.

[56] Arden Dertat. *Applied Deep Learning - Part 4: Convolutional Neural Networks*. - Towards Data Science. Medium. - Visited on March 3, 2023. URL: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

[57] Jason Brownlee. *A Gentle Introduction to k-fold Cross-Validation*. - Machine Learning Mastery. - Visited on March 3, 2023. URL: https://machinelearningmastery.com/k-fold-cross-validation/.

[58] ASTM E2627-13. *Standard Practice for Determining Average Grain Size Using Electron Backscatter Diffraction (EBSD) in Fully Recrystallized Polycrystalline Materials Characterization*. 2013.

[59] David Mainprice et al. "Descriptive tools for the analysis of texture projects with large datasets using MTEX: strength, symmetry and components". In: *Geological Society, London, Special Publications* 409.1 (2015), pp. 251–271.

[60] The GIMP Development Team. *GIMP*. Version 2.10.12. June 12, 2019. URL: https://www.gimp.org.

[61] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[62] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[63] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[64] Paul Gavrikov. *visualkeras*. https://github.com/paulgavrikov/visualkeras. 2020.

Appendices

```python
import numpy as np
import cv2
import os

def Creates_Masks(GB_mask_path, sample_folder,
    grain_size_threshold=500, Open=False,
    remove_grains_in_boundaries=True):
    """
    Creates individual masks for each grain and saves them

    Parameters:

        GB_mask_path (str): path of the grain boundry mask.

        sample_folder (str): path of the sample folder.

        grain_size_threshold (int): mainly used to filter out small
        countours detected due to errors in the grain boundry mask.
        However, it can also be used to filter out small grains.
        The default is 100.

        Open (bool): If True performs Open operation on grain
        boundry mask. It is used to join open lines.

        remove_grains_in_boundaries (bool): If True, does not
        generate masks of grains that lie on the boundry
        of the image.

    """
    gb_mask = cv2.imread(GB_mask_path)
    gb_mask = cv2.cvtColor(gb_mask, cv2.COLOR_RGB2GRAY)
    height, width = gb_mask.shape

    if Open:
        kernel = np.ones((5, 5), np.uint8)
        gb_mask = cv2.morphologyEx(gb_mask, cv2.MORPH_OPEN, kernel)

    contours, hierarchy = cv2.findContours(
        gb_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

    if remove_grains_in_boundaries:
        contours_copy = []
        for i in contours:
            if (0 not in i) and (height - 1 not in i) and (width -
    1 not in i):
                contours_copy.append(i)
```

```
        contours = contours_copy

    object_number = 1
    for i in range(len(contours)):
        if cv2.contourArea(contours[i]) > grain_size_threshold:
            mask = np.zeros((height, width), dtype="uint8")
            cv2.drawContours(
                mask, contours, i, (255, 255, 255),
                thickness=cv2.FILLED)
            file_name = sample_folder[-3:] + "_No_" +
                str(object_number) + ".tif"
            cv2.imwrite(os.path.join(sample_folder, "Masks",
                file_name), mask)
            object_number += 1

    return
```

Source Code 1: Python function that creates masks for each object.

```
import numpy as np
import scipy.io
import cv2
import os

def Apply_Mask(EBSD_array_path, sample_folder, grains = [], pad =
    False, verbose = True):

    """
    Applies masks to the EBSD data

        Parameters:

            EBSD_data_path (str): path of the EBSD matrix.
            sample_folder (str): path of the sample folder.

        Returns:
            grains (list): list with all masked grains.
            Saves the arrays in folder masked_grains.

    """

    f = scipy.io.loadmat(EBSD_array_path)
    globals().update(f) # Creates variable EBSD_array from EBSD
    data file
    height = EBSD_array.shape[0]
    width = EBSD_array.shape[1]
    channels = EBSD_array.shape[2]
```

```
27
28      masks_folder = os.path.join(sample_folder, "Masks")
29
30      for i, file in enumerate(os.listdir(masks_folder)):
31
32          mask = cv2.imread(os.path.join(masks_folder, file))
33          mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
34          mask = cv2.resize(mask, (height, width))
35
36          cnts = cv2.findContours(mask,
37              cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
38          cnts = cnts[0]
39          x, y, w, h = cv2.boundingRect(cnts[0])
40
41          mask = mask[y:y+h, x:x+w]
42          mask = mask / 255.
43
44          masked_array = np.zeros((h, w, channels))
45
46          for j in range(channels):
47              masked_layer = np.multiply(EBSD_array[y:y+h, x:x+w,j],
48                  mask)
49              masked_array[:,:,j] = masked_layer
50
51          file_name = file[:-3] + 'npy'
52          save_path = os.path.join(sample_folder, "EBSD Arrays",
53              file_name)
54          np.save(save_path, masked_array)
55
56          grains.append(masked_array)
57
58      return grains
```

Source Code 2: Python function that applies masks to the full EBSD array to obtain arrays for each object.

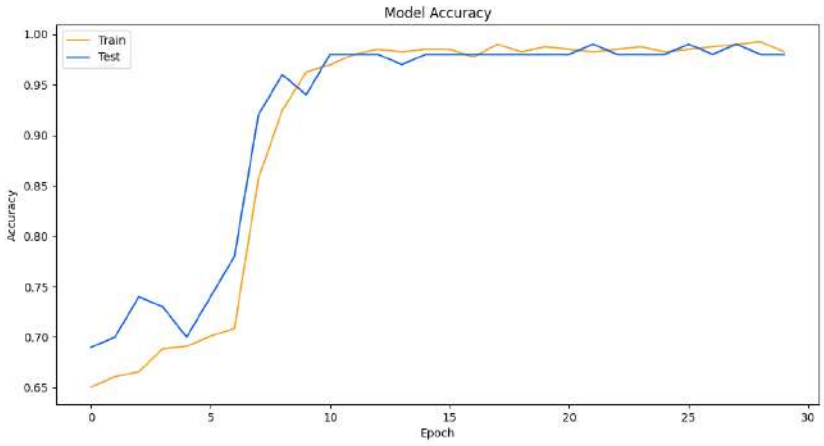

Figure 5.1: Accuracy plot for fold 1 as a function of training epochs for samples A.

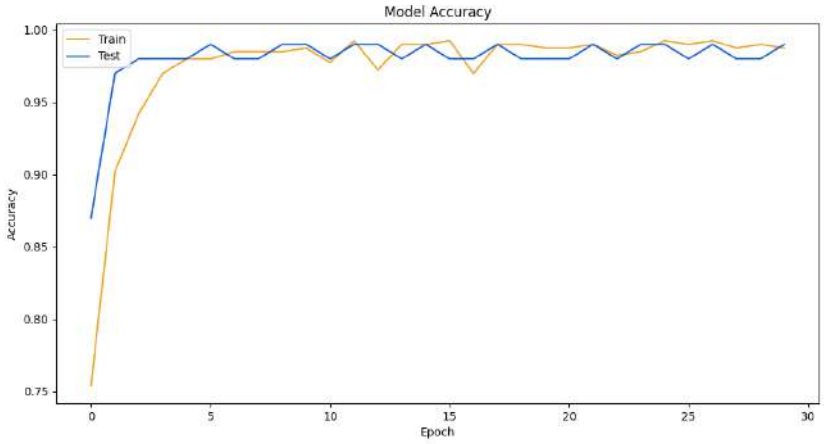

Figure 5.2: Accuracy plot for fold 3 as a function of training epochs for samples A.

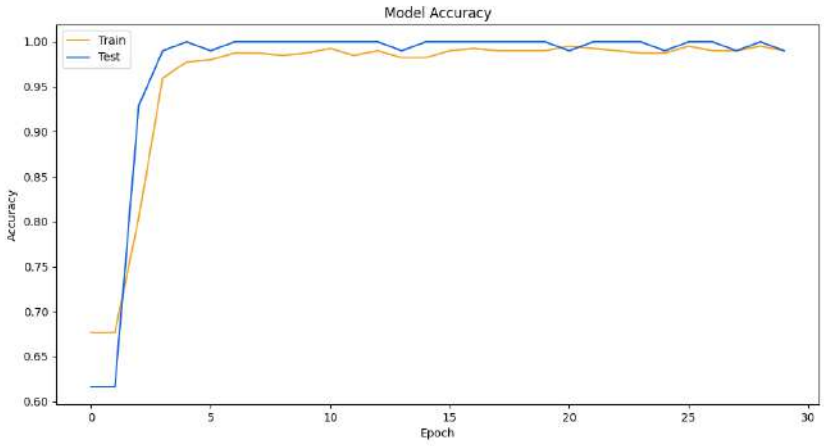

Figure 5.3: Accuracy plot for fold 4 as a function of training epochs for samples A.

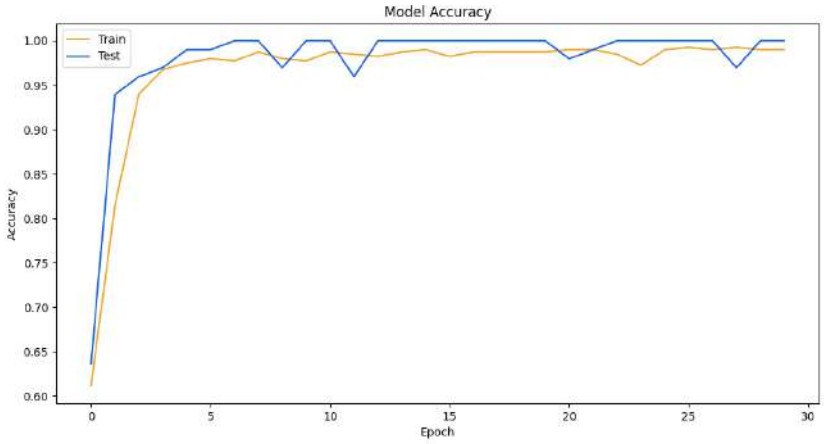

Figure 5.4: Accuracy plot for fold 5 as a function of training epochs for samples A.

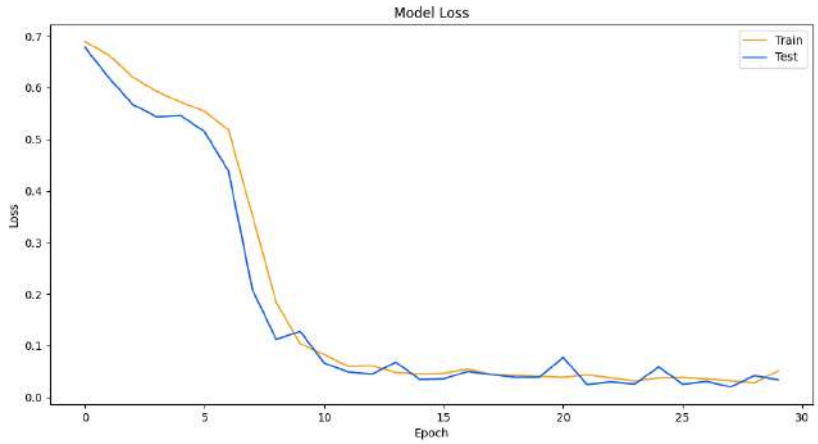

Figure 5.5: Loss plot for fold 1 as a function of training epochs for samples A.

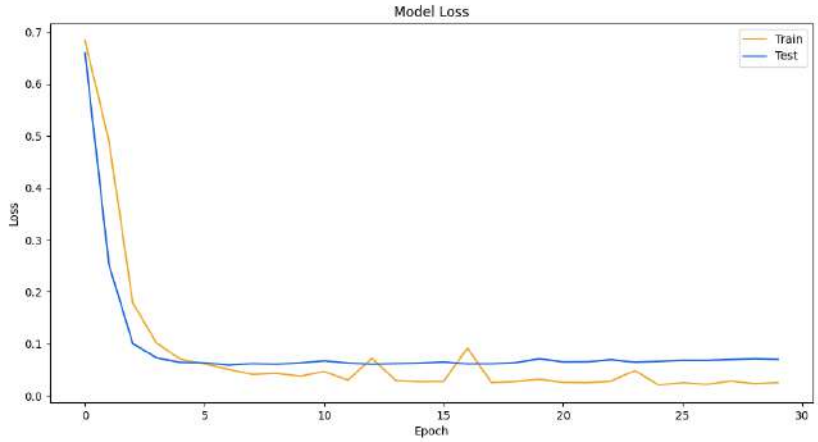

Figure 5.6: Loss plot for fold 3 as a function of training epochs for samples A.

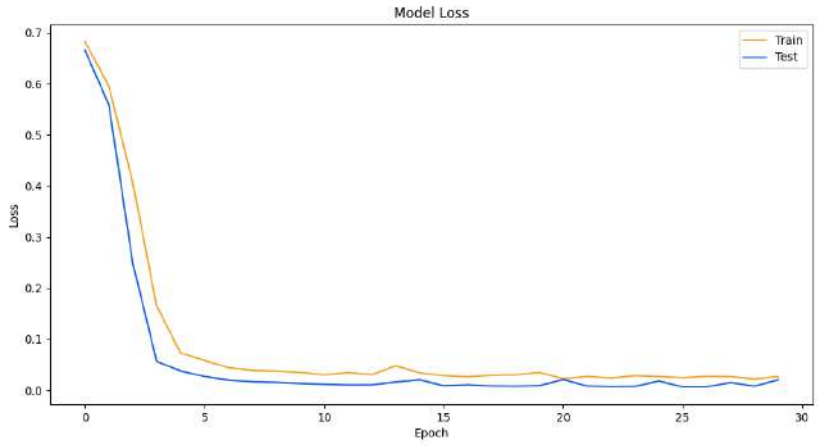

Figure 5.7: Loss plot for fold 4 as a function of training epochs for samples A.

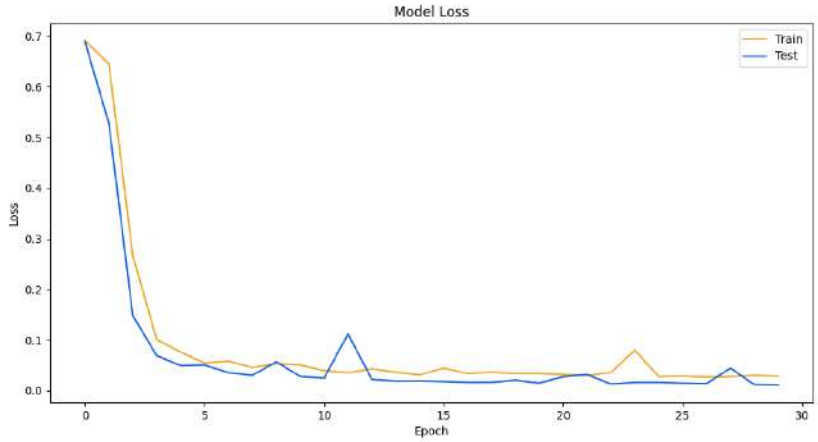

Figure 5.8: Loss plot for fold 5 as a function of training epochs for samples A.

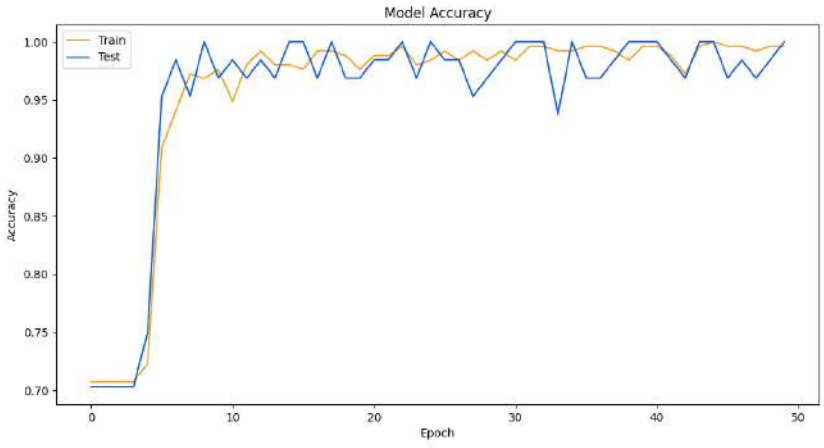

Figure 5.9: Accuracy plot for fold 2 as a function of training epochs for samples C.

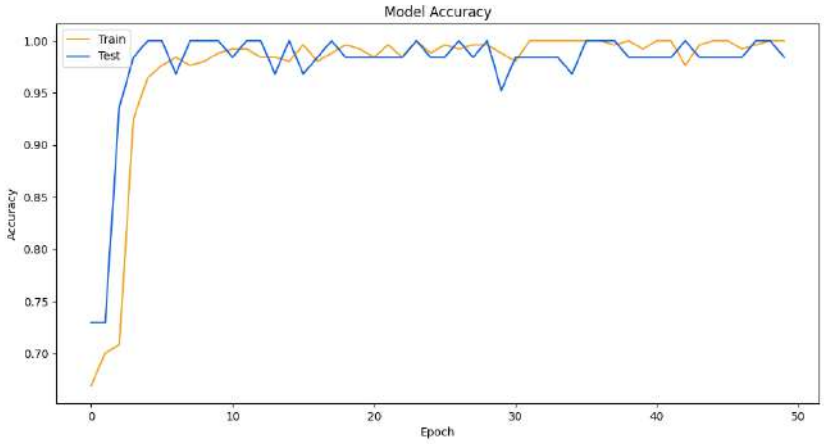

Figure 5.10: Accuracy plot for fold 3 as a function of training epochs for samples C.

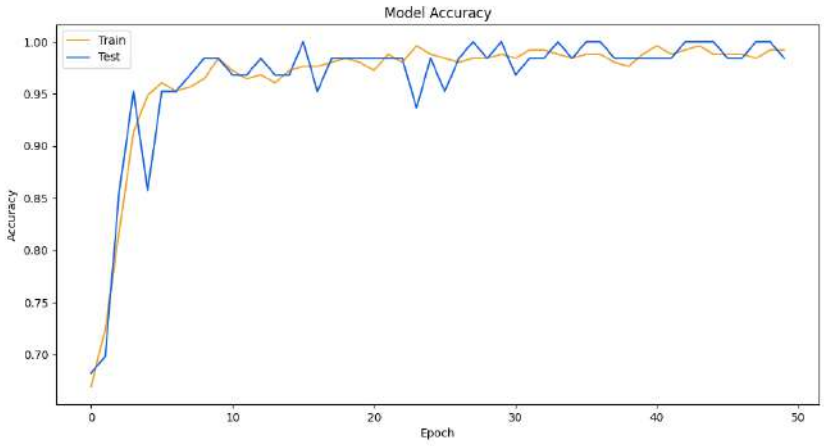

Figure 5.11: Accuracy plot for fold 4 as a function of training epochs for samples C.

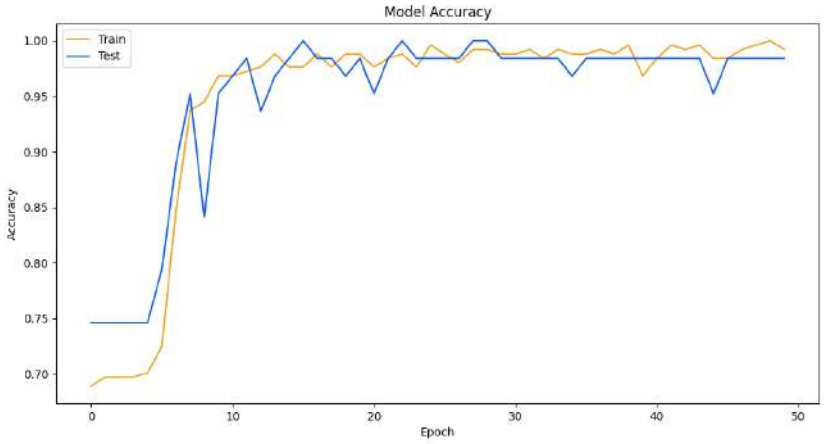

Figure 5.12: Accuracy plot for fold 5 as a function of training epochs for samples C.

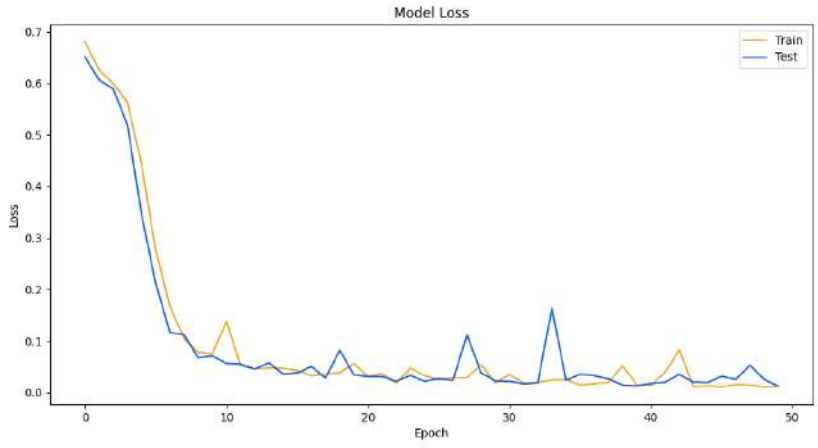

Figure 5.13: Loss plot for fold 2 as a function of training epochs for samples C.

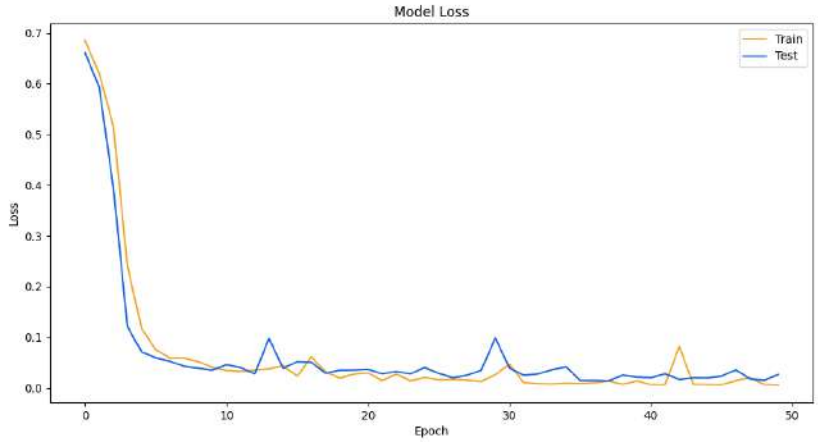

Figure 5.14: Loss plot for fold 3 as a function of training epochs for samples C.

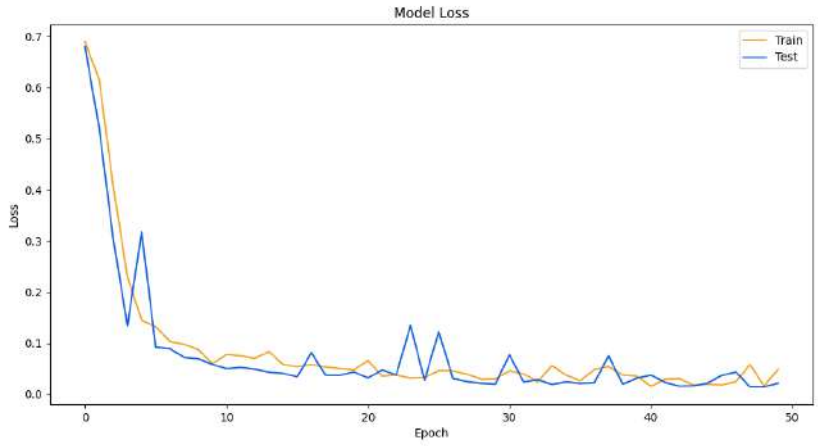

Figure 5.15: Loss plot for fold 4 as a function of training epochs for samples C.

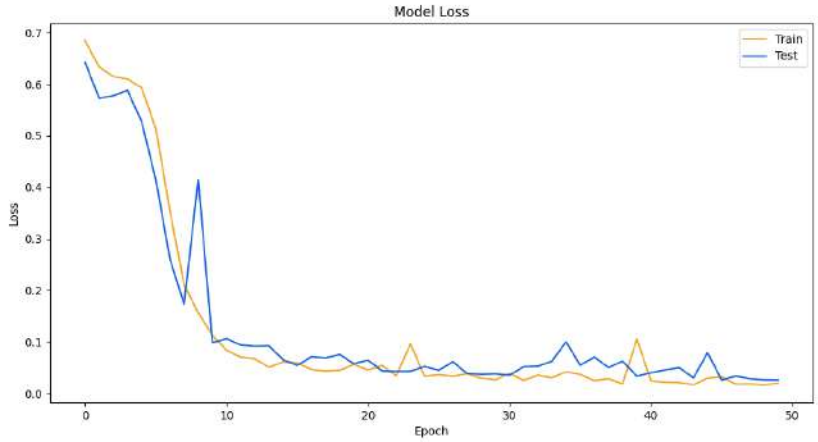

Figure 5.16: Loss plot for fold 5 as a function of training epochs for samples C.