Microstructure property classification of nickel-based superalloys using deep learning

Uchechukwu Nwachukwu¹, Abdulmonem Obaied¹*, Oliver Martin Horst², Muhammad Adil Ali³, Ingo Steinbach¹* and Irina Roslyakova¹*

¹ Interdisciplinary Centre for Advanced Materials Simulation, Ruhr-Universität Bochum, Bochum, 44801, Germany
² Institut für Werkstoffe, Ruhr-Universität Bochum, Bochum, 44801, Germany

E-mail: irina.roslyakova@rub.de

Received 7 June 2021, revised 14 September 2021
Accepted for publication 21 October 2021
Published 5 January 2022

Abstract
Nickel-based superalloys have a wide range of applications in high temperature and stress domains due to their unique mechanical properties. Under mechanical loading at high temperatures, rafting occurs, which reduces the service life of these materials. Rafting is heavily affected by the loading conditions associated with plastic strain; therefore, understanding plastic strain evolution can help understand these material’s service life. This research classifies nickel-based superalloys with respect to creep strain with deep learning techniques, a technique that eliminates the need for manual feature extraction of complex microstructures. Phase-field simulation data that displayed similar results to experiments were used to build a model with pre-trained neural networks with several convolutional neural network architectures and hyper-parameters. The optimized hyper-parameters were transferred to scanning electron microscopy images of nickel-based superalloys to build a new model. This fine-tuning process helped mitigate the effect of a small experimental dataset. The built models achieved a classification accuracy of 97.74% on phase-field data and 100% accuracy on experimental data after fine-tuning.

* Author to whom any correspondence should be addressed.

Original content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Keywords: deep learning, computer vision, Ni-based superalloys, classification, microstructure

(Some figures may appear in colour only in the online journal)

1. Introduction

Nickel-based superalloys are widely used for high temperature, high pressure, and high-stress applications due to their excellent mechanical properties, which include good fatigue and creep resistance even at severe temperature and high mechanical strength [1, 2]. The inception of these excellent mechanical properties emanates from their two-phase microstructure of \( \gamma \) disordered matrix phase and \( \gamma' \) precipitate phase having L12 crystal structure. The ordered \( \gamma' \) phase, with a volume fraction of about 60%–70%, acts as a strengthening phase, suppressing dislocation movement and atomic diffusion during creep deformation [3–5].

Due to its microstructure, its usage in the aerospace industries and power generation plants are becoming increasingly famous, ranging by weight to about 50% of turbine engines in aircrafts [6]. Further development and improvement have led to the higher efficiency of turbine engines, having a significant impact on environmental emissions. Their application niche subjects them to continuous exposure to mechanical loading at high temperatures. This induces directional coarsening called rafting, creep deformation, and microstructural evolution, leading to a reduction in the mechanical properties, resulting in the reduced service life of the material [7, 8]. In addition, rafting increases the plastic strain in corresponding gamma channels (\( \gamma \)) and interphases by widening the gamma-prime channels (\( \gamma' \)) range. As a result, it is crucial to analyze the microstructural evolution following creep failure using a creep test with increasing time and strain. Image analysis techniques in materials science, which advances in computers and electronic devices have enhanced, may aid in understanding the relationship between structure and functional relationships in materials, and thus microstructural evolution. Mansur et al successfully used machine learning methods in conjunction with image processing to investigate the microstructural evolution of nickel-based superalloys [9]. However, due to the extreme complexity of microstructural features, it is often difficult to understand the various parameters that play specific roles in the evolution following creep failure. Deep learning has been shown to learn the features for microstructural recognition from unprocessed input images alongside the classification task, thus eliminating the need for manual feature engineering.

This research aims to study and understand the alloys microstructure with respect to specific parameters such as strain using deep learning techniques, as several investigations have revealed that rafting occurs due to loading conditions associated with plastic strain and high-temperature fatigue loading [10].

This research has the following structure. First, a brief description of computer vision and deep learning applications in materials science, followed by examining the most commonly used deep learning techniques and architectures for microstructural image classification. Finally, deep learning techniques are applied to our current task, and the results are analyzed.

2. Computer vision and deep learning

The role of micro-structural images in the growth of materials science cannot be overemphasized, as they play a critical role in the understanding of materials and groups of microstructures through feature analysis. Proper interpretation of digital microstructural
images can help to identify features to support segmentation, characterization, and high precision comparison of microstructures [11]. The field of computer vision, which is rapidly evolving and gaining a lot of attention in recent times due to its accomplishments in interpreting and extracting digital image information, can be extended to the interpretation of microstructural images. Computer vision can be defined as developing algorithms and models to comprehend and recognize information in images by modeling the human visual system’s complexity [12].

Computer vision’s growth has greatly been enhanced by deep learning, a branch of machine learning inspired by the human brain’s structure and function, consisting of neurons that perform basic operations and coordinate with one another before a decision is made [12]. Deep learning’s success in solving complex problems stems from its ability to learn complicated structures in complex data with little or no human intervention [13]. Traditional machine learning techniques required careful human engineering with domain expertise and knowledge to extract meaningful features from raw data, whereas deep learning approaches automatically learn these features from the raw data [14].

In recent years, the convolutional neural network has been the most suitable algorithm for solving computer vision tasks due to its accuracy in handling image tasks. CNN LeNet-5 was first illustrated by LeCun (figure 1) in 1998 for digit recognition [15]. Still, it was not until 2012 that CNN gained prominence when AlexNet emerged as the winner of the ImageNet ILSVRC competition boasting a 15.4% error rate, besting the next runner up with a 10.8% difference [16]. AlexNet was built using the same pattern of LeNet-5 architecture. This renewed interest led to numerous research efforts in this field, leading to an improved version, ZFNet, with the same structure as AlexNet to be the ILSVRC winner in 2013 [17]. The Inception and VGG-16 architectures further demonstrated the accuracy of CNN’s improved with increasing layers, with Inception winning the ILSVRC 2014 competition and VGG-16 finishing first runner-up [18, 19]. Further improvements in CNN accuracies were made with the introduction of skip connections, making the possibility of deeper networks with lower complexity possible. ResNet, built with skip connections, had 152 layers and emerged victorious in the 2015 ILSVRC challenge [20].

The growth of deep learning algorithms has also been extended to materials science, with Azimi et al showing the classification of steels into martensite, bainite, and perlite [21], Chowdury et al using deep learning to recognize dendritic morphologies [22], and Iglesias et al using deep learning for optical micrographs [23]. Feng et al also used deep learning to
predict material defects with limited datasets [24]. Deep learning networks were utilized in several works of Lin et al. to forecast the hot deformation behavior of alloys and analyze the hot deformation behavior of Al–Zn–Mg–Cu and Ni-based superalloys [25, 26]. Chen et al. used neural networks in conjunction with cellular automaton simulation to develop a material design framework that successfully optimizes processing parameters for target microstructures, resulting in fine and uniform Ni-based superalloys [27], while Lin et al. investigated hot compression behaviors and microstructures using artificial neural networks [28].

This study aims to apply deep learning methods to the microstructural classification of creep strain, overcoming the challenge of limited data by training deep layers with phase-field simulation data, which has been shown to exhibit similar results with experiments and then transferring already gained knowledge from the simulated data to solve our classification task on experimental data.

2.1. Convolutional neural networks

The most well-known deep learning algorithm is the convolutional neural network, a type of artificial neural network that has demonstrated exceptional success in pattern recognition and image processing [29]. They capture the spatial information between an image’s pixels by applying relevant filters. The components of CNN are outlined in the next section.

2.2. Features of convolutional network

**Convolution layer:** the convolution operation can be described as the combined integration of two functions, illustrating how one function modifies the other. The convolution layer is a type of linear operation that performs distinct feature extraction of the input image by applying filters to generate a feature map, as shown in figure 2.

The filters, also known as kernels, detect different patterns in the images and preserves the spatial relationship between pixels. The complexity of features to be extracted increases with increasing layers, with the highest level features extracted in the final layers. Equation (1) describes the convolution (∗) of the input data with respect to the filters:

\[(h_k)_{ij} = (W_k \ast x)_{ij} + b_k,\]  \hspace{1cm} (1)

where \((h_k)_{ij}\) specifies the neuron’s output in the \(k\)th feature map for the position \((i, j)\), \(k = 1, \ldots, K\) is the index in the convolution layer corresponding to the \(k\)th feature map, \(x\) denotes the input data, \(W_k\) indicates the weight, \(b_k\) denotes the bias for the \(k\)th feature map [21].
Pooling layer: the pooling layers perform a non-linear downsampling operation, reducing the number of feature map’s parameters, making it vigorous against distortions and noise. This leads to a reduction of overfitting and saves computational cost. The types of pooling are max pooling and average pooling shown in figure 3. Max pooling is commonly used as it outperforms in preserving the spatial invariance in the image by returning the maximum value from a patch of input features [30]. The hyper-parameters of the pooling layer are the filter size and the stride, which are both recommended to be of size $2 \times 2$.

Fully connected layers: the output of the pooling layer is flattened and passed to the fully connected layer, which is similar to the generic neural network. This layer combines features for classification with a linear equation indicated in equation (2). The weights are adjusted in this layer

$$ y_k = \sum_j W_{kl} x_j + b_k, $$

where $y_k$ is the $k$th output neuron, $W_{kl}$ indicates the $kl$-th weight within $x_i$ and $y_k$ [21].

Activation functions: CNN employs the activation function in the network to boost the desirable non-linearity, detect non-linear features, and learn more complex models. The ReLu activation function $\text{ReLu}(x) = \max(0, x)$ was used as it is easy to train and speeds up training.

Classification layer and loss function: the final layer contains the output class of the input image. A softmax function denoted by equation (3) is used to force the output to represent a probability distribution across discrete alternatives

$$ P(y = j | X; W, b) = \frac{\exp^{x^T W_j}}{\sum_k \exp^{x^T W_k}}, $$

where $y$ is the input vector, and a categorical probability distribution for the $j$th class and input vector of $X$ is denoted by a vector of values within $(0, 1)$ through this function [21].

The loss function is used to optimize the parameters to maximize the architecture’s effectiveness. The cross-entropy function $H$ shown in equation (4) is used as it takes the function’s logarithm to notice even slight improvements during backpropagation

$$ H = -\sum_x P(x) \log P(x), $$

\[5\]
where \( P(x) \) reduces the cross-entropy of correct distribution of data \( P'(x) \) and predicts class probabilities, where \( P'(x) \) is 1 for true class and 0 for others [21].

2.3. Training a CNN network

Training CNN is a global optimization problem involving determining the best fitting parameters by minimizing the loss function. The input data, an image, is propagated forward through the model to produce an output with an initial set of arbitrary parameters. The loss on the output is then determined using the loss function, which evaluates the discrepancies between the predicted label and the true label of the input for a given set of values. The loss function is reduced using stochastic gradient descent, an iterative optimization algorithm that calculates the gradient of the loss function by backpropagation and updates the model’s weights. Training can be improved by repeatedly passing the data through the model with each pass through the network known as an epoch. The learning rate determines how the weights are modified.

2.4. CNN architectures

Architecture involves how the layers in the neural network are arranged. The architectures employed with the available computational resources are the AlexNet and ResNet architectures.

2.4.1. AlexNet. AlexNet, proposed by Krizhevsky et al., is regarded as one of the first CNN architectures to achieve outstanding results in image classification tasks by deepening CNNs and employing a variety of parameter optimization techniques. AlexNet architecture consists of eight layers shown in figure 4, five convolutional layers, and three fully connected layers. It employs the ReLu activation function to mitigate the effect of vanishing gradients and increase the convergence rate. The model consists of 60 million parameters and 650,000 neurons.

2.4.2. ResNet. ResNet architecture was proposed by He et al., introduced residual learning, and helped solve the vanishing gradient problem that arises as our networks become deeper, resulting in accuracy loss. Skip connections are introduced to provide identity mapping, properly described in [20], which allows stacking of deeper networks without adding additional parameters or computational complexity hence avoiding the degradation in accuracy.

3. Methods

In this section, we illustrate the model architectures and approaches taken to classify nickel-based superalloys with reference to their creep strain levels. First, we acquire the datasets for the simulated and experimental images then perform dataset pre-processing and augmentation. Furthermore, we implement deep learning models to the simulated dataset with various parameter values, obtaining an optimized model whose weights and parameters would be transferred to the experimental data set by fine-tuning. Finally, a model is built using the fine-tuned values with the experimental dataset.

3.1. Dataset

The dataset investigated comprises phase-field simulated images and SEM nickel-based superalloys experimental images. The phase-field dataset was acquired from the STKS department [31–34], through a simulation with an applied stress of 350 MPa and temperature of 950°C shown in figure 6. The simulation attained similar creep strains to that of experiments. The
Simulated Creep Test

External Stress = 350 MPa
Temperature = 950°C

![Simulated creep test compared with experiment performed under same conditions](image)

Figure 4. Simulated creep test compared with experiment performed under same conditions [51].

Phase-field simulation microstructure consists of creep strain levels from 0.0%–1.0% at different interval steps. A total of 128 slices were cut from the microstructure in each step in the x, y, and z-direction, producing 10752 images. The images were then cropped to 390 × 390 pixels to eliminate regions without relevant information (figure 7). The images were sorted according to the classes of creep strain levels ranging from 0.0%–0.4%, 0.4%–0.6%, 0.6%–0.8%, and 0.8%–1.0%.

The experimental images were provided by researchers from the Lehrstuhl für Werkstoffwissenschaft, Institute for Materials, Ruhr-University Bochum [35, 36]. The images were obtained from dendritic regions in ERBO/1 single-crystal nickel-based superalloy with varying strain levels, as shown in figure 8. A total of 15 images ranging from prior creep to 0.4%, 0.6%, 1.0%, and 2.0% strain, were obtained by conducting disrupted tensile creep tests in the [001] direction at 950°C and 350 MPa. All images were generated using the SEM technique at a magnification of ×10000 and transformed to grayscale images with eight-bit depth, having 1024 × 768 pixels. The experimental images were sorted into strain levels ranging from 0.0%–0.4%, 0.4%–0.6%, 0.6%–1.0% and greater than 1.0%.

The SEM images used to train the model further had similar creep strain results to the simulated data under the same temperature and stress conditions.

3.2. Data preprocessing and augmentation

The images of both classes were split into training and testing data randomly at an eight to two ratio. The training data was used to select and build the model parameters while the test data was used after training to evaluate the model’s performance, simulating the deployment of our
model in real-world unknown cases. The random initialization is to help the generalization of the model so as not to limit the model to learn from only specific instances.

Data augmentation was used to compensate for the limited availability of the experimental datasets, as we were unable to obtain additional images online because the majority of nickel-based superalloy images found online were unlabeled, i.e. without the proper strain levels. Data augmentation involves applying different approaches to improve the size and quality of datasets helping to solve the problem of limited data [37]. Data augmentation was applied to the images to expand the variety of data available to train the models involving the transformation of the images by flipping horizontally and vertically, zooming, rotating based on angles, and intensity variation for brightening and darkening of the images shown in figure 9. Data augmentation boosts model regularization by expanding the training image dataset, which improves the model’s generalization and reduces the overfitting problem. The images were resized to specific sizes tailored to the model’s architectures.

3.3. Implementation details

The model was built in Python programming language (https://python.org, version 3.8.5) [38] using PyTorch (https://pytorch.org, version 1.7.0) [39] and FastAI (version 2.1.8) [40] libraries with GPU ‘GeForce GTX 960’. FastAI is a deep learning library that provides
Figure 6. Data augmentation performed on (a) simulated and (b) experimental datasets.

Figure 7. Training and validation curves for (a) batch size 16 without data augmentation (b) batch size 16 (c) batch size 32 (d) batch size 64.

researchers with high-level components to produce cutting-edge results in traditional deep learning domains [41]. FastAI has exhibited excellent results in numerous classification tasks in various research niches including materials science [42] and medicine [43, 44].
3.4. Implementation details for simulated images

The classification of simulated images was done with several convolutional neural networks of varying architectures, limited to the computational capacity (ResNet18, ResNet34, and AlexNet). Transfer learning which uses already gained knowledge from previously trained models to solve a new task, was used to build the model. Our current task uses pre-trained ConvNets from PyTorch libraries that have been trained on millions of images using the open-source ILSVRC [45]. Due to the dissimilarities between the current dataset and the dataset used for training, the layers in the pre-trained model were unfrozen, making it possible for them to be trained and updated. A differential learning rate was employed for training as final layers are more probable to require additional training. The batch size was alternated between 16, 32, and 64 due to the GPU RAM constraint, and the model predictions were evaluated using the metric ‘error-rate’ which is introduced in the later sections. The saved weights were saved and exported as a pickle Python file after training to be employed on the experimental images.
Table 1. Confusion matrix for two-class classification.

| Predicted positive | Actual positive | TPs | Actual negative | FNs | Predicted negative | FPs | TNs |
|--------------------|-----------------|-----|-----------------|-----|--------------------|-----|-----|

3.5. Implementation details for experimental images

The classification model for experimental images was fine-tuned by using the previously saved weights from the simulation model. This helps to mitigate the lack of experimental images as some similar features from simulation images have already been learned and can be applied to experimental data. This provides better initialization of parameters, which aids in model generalization, reducing overfitting, decreasing training time, and improving accuracy. This model was created using the simulation model’s optimized architectures and hyper-parameters.

4. Performance evaluation

After implementing and training our model, it is necessary to evaluate the model’s performance to determine how accurate the model learned from the data. Metrics are employed in the evaluation of the model’s predictions and can be used to compare different models. The error rate that estimates the number of misclassified instances with respect to the total instances available is the key metric used. The major metrics used for classification can be implemented with the confusion matrix, which is introduced below.

4.1. Confusion matrix

A confusion matrix summarizes the model’s performance with respect to the test data for which label values were known. Table 1 shows a confusion matrix.

Here TPs are values that were correctly predicted to be positive, while TNs are values that were accurately predicted to be negative, FPs are values wrongly predicted as positive, and FNs are values wrongly predicted as negative. The confusion matrix diagonal represents the correctly predicted cases. The accuracy, precision, recall (sensitivity), specificity, F1, and error rate metrics can be calculated using the confusion matrix.

**Accuracy** indicates the total number of samples correctly classified in comparison to the total number of samples

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5}\n\]

**Precision** is defined as the proportion of true positive values to the total number of positive values

\[
\text{Precision} = \frac{TP}{TP + FP}. \tag{6}\n\]

**Recall (sensitivity)** is a measure of how many true positive values have been classified correctly out of the total number of positive values

\[
\text{Recall} = \frac{TP}{TP + FN}. \tag{7}\n\]
Table 2. Strain classification results using AlexNet architecture.

| Network  | Augmentation | Batch size | Error rate  |
|----------|--------------|------------|-------------|
| AlexNet  | No           | 16         | 0.052775    |
| AlexNet  | Yes          | 16         | 0.081610    |
| AlexNet  | Yes          | 32         | 0.066104    |
| AlexNet  | Yes          | 64         | 0.068281    |

Specificity represents the true negative rate

\[
\text{Specificity} = \frac{TN}{TN + FP}.
\]

F1 is the harmonic means of precision and recall

\[
F1 = \frac{2TP}{2TP + FP + FN}.
\]

Error rate ER is indicated by

\[
ER = 1 - \text{Accuracy}.
\]

The metrics error rate and precision are the essential measures employed as they reflect the ratio of misclassified classes and produce values that can easily be interpreted and used.

5. Results and discussions

Firstly, we discuss the findings acquired when training the CNN with the simulation dataset and then present the results achieved by applying this method to the experimental dataset.

5.1. Evaluation of data augmentation

The effect of data augmentation on the images was analyzed by comparing the model’s accuracies with and without data augmentation. For the phase-field images, it was noticed that the model performed slightly better without augmentation with an accuracy of 97.74% against 97.63% without applying data augmentation for a batch size of 16. The model could only be trained with a batch size of 16 without augmentation as part of the included augmentation resizes the images in the batch before being passed to the model.

The effect of augmentation on the experimental images was more profound as it helped reduce the training and validation loss. Similar accuracies were achieved in both cases.

5.2. Evaluation of network architecture

The architectures used in this report were AlexNet, ResNet18, and ResNet34. Due to the limitation of RAM, additional networks could not be used. Tables 2–4 shows the different architectures used and their error rate for training five epochs each.

From our observations and results presented in tables 2–4, the ResNet34 network was found to be the most suitable model for the classification task as it outperformed the other networks in each of the comparative cases.
Table 3. Strain classification results using ResNet18 architecture.

| Network  | Augmentation | Batch size | Error rate   |
|----------|--------------|------------|--------------|
| ResNet18 | No           | 16         | 0.030740     |
| ResNet18 | Yes          | 16         | 0.044614     |
| ResNet18 | Yes          | 32         | 0.039173     |
| ResNet18 | Yes          | 64         | 0.042437     |

Table 4. Strain classification results using ResNet34 architecture.

| Network  | Augmentation | Batch size | Error rate   |
|----------|--------------|------------|--------------|
| ResNet34 | No           | 16         | 0.024211     |
| ResNet34 | Yes          | 16         | 0.034276     |
| ResNet34 | Yes          | 32         | 0.035637     |
| ResNet34 | Yes          | 64         | 0.034820     |

5.3. Evaluation of hyper-parameters

The efficiency of the model is highly dependent on the model’s parameters and hyper-parameters. Model parameters are model-dependent parameters whose values are learned during training. On the other hand, hyper-parameters cannot be learned directly from training the model and must be supplied to the model before the training process begins. Hyper-parameters govern the entire structure of the model; thus, selecting the right hyper-parameters is critical for the model’s accuracy. The hyper-parameter selection is often unique to different tasks. Optimum hyper-parameters are obtained by training different models with various hyper-parameter values and deciding the most accurate values by evaluating the different models.

The hyper-parameters used in this model include batch size, learning rate, and loss function.

5.3.1. Batch size. The batch size specifies the number of samples that must pass through the network before the model variables are updated. Large batch sizes can lead to the model’s generalization, while small batch sizes can lead to several iterations before the update of the model. Implementation of batching helps us reduce computational memory required to train our model and helps train the model faster. For better regularization, smaller batch sizes are recommended [46], as it helps improve accuracy. Due to computational memory, batch sizes greater than 64 could not be implemented. Table 5 shows the results of the different batch sizes with the ResNet34 architecture, using cyclic learning [47]. The different batch sizes produced similar accuracies. Batch size of 16 without data augmentation and the batch size of 32 produced the highest accuracies. A plot of training and validation loss used to evaluate the model’s fitting is shown in figure 10. With the exception of batch size 16 without data augmentation, the runtime for the execution of one epoch in seconds was found to decrease with increasing batch size.

5.3.2. Learning rate. The learning rate hyper-parameter governs the model’s response to the to the estimated error whenever the model weights are adjusted [48]. Low learning rates result in more efficient training but slower optimization, while large learning rates can result in diverging as the optimizer can overshoot the accurate minimum. The learning rate $\eta$ can be gotten
Table 5. Classification results with respect to a batch size of ResNet34 architecture.

| Batch size | Data augmentation | Network    | Training loss | Validation Loss | Error rate | time/s |
|------------|-------------------|------------|---------------|-----------------|------------|--------|
| 16         | No                | ResNet34   | 0.039441      | 0.052183        | 0.022579   | 711    |
| 16         | Yes               | ResNet34   | 0.079830      | 0.077069        | 0.026115   | 346    |
| 32         | Yes               | ResNet34   | 0.058640      | 0.062838        | 0.023667   | 312    |
| 64         | Yes               | ResNet34   | 0.055197      | 0.066442        | 0.024755   | 301    |

Figure 10. Training and validation curves for (a) batch size 16 without data augmentation (b) batch size 16 (c) batch size 32 (d) batch size 64.

from equation (11)

\[ w^n = w^{old} - \eta \frac{\delta \mathcal{L}}{\delta w} \]  

where \( w \) is the weights and \( \mathcal{L} \) is the loss function. The choice of learning rate is a very important parameter for training our model, and the FastAI method ‘learning rate finder’ helps to give a good estimate for an optimal learning rate. Figure 11 shows the plot of loss versus learning rate, with appropriate learning rate values observed prior to the curve’s minimum. The learning rate value at the minimum point is not chosen as it is too large, and the model would not be able to converge. A learning rate of \( 1 \times 10^{-3} \) was employed in training the model with frozen layers, while a cyclic learning rate was implemented after unfreezing the weights to enable the learning rate to change within a range of values, thus helping the model converge faster and improve accuracy [47]. The used cyclic rate gotten from the learning rate finder suggestion was between \( 1 \times 10^{-9} \) to \( 1 \times 10^{-2} \). Table 6 shows the comparison of fixed learning rates on frozen layers and cyclic learning rates after unfreezing. The addition of cyclic learning rates was found to minimize the percentage errors alongside the training and validation loss for all batch sizes.
5.3.3. Loss function. The loss function is used to optimize the model by evaluating the error value, which describes how the model’s output varies from the expected output [21]. The stochastic gradient descent method is used to minimize the loss function by back-propagating the error to the first layer and updating the model weights at each iteration. The gradients are calculated through backpropagation [49]. The cross-entropy loss function is defined by equation (4) and label smoothing cross-entropy is defined by

\[ P'(x)^{LS} = P'(x)(1 - \alpha) + \frac{\alpha}{K}, \]  

(12)

where \( \alpha \) is the label smoothing parameter and \( K \) is the number of label classes. Both loss functions were applied in the model, with cross-entropy achieving greater accuracies, and less training and validation loss.

5.4. Computational runtime

Due to the computational intensity of deep learning models, evaluation of computational time and memory is essential for limited computational resources. Figure 12 shows the computational time for one epoch training for different architectures with respect to the batch sizes and
Figure 12. Runtime comparison for training one epoch of different architectures with respect to batch sizes (a) fixed learning rate (b) cyclic learning rate.

Table 7. Confusion matrix of phase-field simulated data without data augmentation.

| Actual predicted | 0.0%–0.4% | 0.4%–0.6% | 0.6%–0.8% | 0.8%–1.0% | Class precision |
|------------------|-----------|-----------|-----------|-----------|----------------|
| 0.0%–0.4%        | 919       | 11        | 0         | 0         | 98.82%         |
| 0.4%–0.6%        | 0         | 920       | 3         | 0         | 99.67%         |
| 0.6%–0.8%        | 0         | 17        | 882       | 16        | 96.39%         |
| 0.8%–1.0%        | 0         | 5         | 31        | 872       | 96.04%         |
| Class recall     | 100%      | 96.54%    | 96.29%    | 98.20%    | Accuracy: 97.74% |

Table 8. Confusion matrix of phase-field simulated data with data augmentation.

| Actual predicted | 0.0%–0.4% | 0.4%–0.6% | 0.6%–0.8% | 0.8%–1.0% | Class precision |
|------------------|-----------|-----------|-----------|-----------|----------------|
| 0.0%–0.4%        | 923       | 7         | 0         | 0         | 99.25%         |
| 0.4%–0.6%        | 0         | 921       | 2         | 0         | 99.78%         |
| 0.6%–0.8%        | 0         | 23        | 863       | 29        | 94.32%         |
| 0.8%–1.0%        | 0         | 5         | 21        | 882       | 97.14%         |
| Class recall     | 100.00%   | 96.34%    | 97.40%    | 96.82%    | Accuracy: 97.63% |

learning rates. The training time increases with the depth of the architecture stemming from the fact that deep networks are harder to train. Increasing batch size helps improve parallelization in GPUs, thereby making training faster for ResNet networks.

The effect of increasing batch size in the AlexNet network was found to be varying. Cyclic learning rates were observed to increase the training time for one epoch as it cycles through a range of values, but this helps the model converge faster.

Tables 7 and 8 shows the confusion matrix of the most accurate models built using phase-field datasets with and without data augmentation. The majority of misclassified instances were found to be in the neighboring classes.

5.5. Experimental dataset results

The optimized parameters obtained after hyperparameter tuning were applied to the experimental data by fine-tuning. Data augmentation helped reduce the validation and training loss as similar accuracies were achieved with and without augmentation.
Table 9. Confusion matrix of SEM nickel-based superalloy images with fine-tuning technique showing a 100% accuracy.

| Actual predicted | 0.0%–0.4% | 0.4%–0.6% | 0.6%–1.0% | >1.0% | Class precision |
|------------------|-----------|-----------|-----------|-------|-----------------|
| 0.0%–0.4%        | 18        | 0         | 0         | 0     | 100.00%         |
| 0.4%–0.6%        | 0         | 23        | 0         | 0     | 100.00%         |
| 0.6%–1.0%        | 0         | 0         | 24        | 0     | 100.00%         |
| >1.0%            | 0         | 0         | 0         | 23    | 100.00%         |
| Class recall     | 100.00%   | 100.00%   | 100.00%   | 100.00% | Accuracy: 100.00% |

Table 10. Confusion matrix of the SEM nickel-based superalloys without fine-tuning showing a 98.86% accuracy.

| Actual predicted | 0.0%–0.4% | 0.4%–0.6% | 0.6%–1.0% | >1.0% | Class precision |
|------------------|-----------|-----------|-----------|-------|-----------------|
| 0.0%–0.4%        | 18        | 0         | 0         | 0     | 100.00%         |
| 0.4%–0.6%        | 1         | 22        | 0         | 0     | 95.65%          |
| 0.6%–1.0%        | 0         | 0         | 24        | 0     | 100.00%         |
| >1.0%            | 0         | 0         | 0         | 23    | 100.00%         |
| Class recall     | 94.74%    | 100.00%   | 100.00%   | 100.00% | Accuracy: 98.86% |

Overfitting is a fundamental problem in deep learning that prevents the model’s effective generalization from observed to unseen data. It was handled in this model by applying several techniques to reduce the effect of overfitting. Data augmentation was used to increase the amount of available data to help the generalization of the model. Pre-trained networks also helped model generalization by initially training a model on a more extensive training set with a similar domain. Cyclic learning rates helped improve the model’s convergence with fewer training iterations, preventing some possibility of learning unwanted parameters and noise. Choosing an appropriate loss function helped the model’s regularization.

Tables 9 and 10 shows the confusion matrix of the experimental images with and without fine-tuning. A 100% accuracy was achieved with fine-tuning, classifying all instances correctly, while a 98.86% accuracy was achieved without fine-tuning.

5.6. Testing on literature images

The model was tested on images gotten from works of literature that were not used to train the model. The images only indicated the rafting condition and not the percentage strain and were all dendritic images of nickel-based superalloy.

The predicted outcomes are shown in table 11. The source column in table 11 specifies the papers and figures from which the images were obtained. The images were cropped into separate images when the figures included more than one image. The microstructure of CMSX-4 before and after rafting is depicted in table 11 images 1 and 2. Table 11 images 3 and 4 show SEM images cropped images of DD5 before and after rafting. Table 11 images 5 and 6 show unrafted CMSX-4 superalloy and rafted superalloy, respectively, with image 5 representing the as-received microstructure and image 6 representing N-type rafterd images. Both photos were cropped from a larger image. TEM images of SRR99 superalloy are shown in table 11 images 7–9.

From table 11 applying our model to image 1 from source figure 1 [3] we obtained that this microstructure corresponds to the initial state of the CMSX-4 superalloy with a probability of 62.89% for the class of 0.0%–0.4%. Image 2 from source figure 2 [3] corresponds with the
Table 11. Experimental image conditions and predicted strain results.

| Image Source | Alloy | Temp. (°C) | Stress (MPa) | 0%–0.4% | 0.4%–0.6% | 0.6%–1.0% | >1.0% |
|--------------|-------|------------|--------------|---------|-----------|-----------|-------|
| Figure 1 [3] | CMSX-4 | N/A        | N/A          | 62.89   | 35.00     | 0.02      | 2.09  |
| Figure 2 [3] | CMSX-4 | 2.41       | 1000         | 0.04    | 0.00      | 0.12      | 99.83 |
| Figure 1(a) [4] | DD5 | Initial state | N/A | 100.00 | 0.00 | 0.00 | 100.00 |
| Figure 1(b) [4] | DD5 | After rafting | 980 | N/A | 0.00 | 0.01 | 0.01 | 99.98 |
| Figure 1 [52] | CMSX-4 | Initial state | 1050 | 6.25 | 81.41 | 9.90 | 2.43 |
| Figure 1N-type [52] | CMSX-4 | After rafting | 1050 | N/A | 0.00 | 0.07 | 99.93 |
| Figure 2(a) [53] | SRR99 | Initial state | 980 | 2.50 | 6.73 | 70.09 | 9.44 | 13.74 |
| Figure 2(b) [53] | SRR99 | After rafting | 980 | 200 | 0.00 | 0.00 | 0.00 | 100.00 |
| Figure 2(c) [53] | SRR99 | After rafting | 980 | 200 | 0.00 | 0.00 | 0.00 | 100.00 |
rafted microstructure of CMSX-4 showing a probability of 99.83% for the > 1.0% class. Image 3, which was cropped out from source figure 1(a) [4] corresponds with the initial microstructure prior deformation of DD5 superalloy with a probability of 100% for the 0.0%–0.4% class and image 4 cropped out from source figure 1(b) [4] having a probability 99.98% for the > 1.0% class corresponds to a rafted structure after creep deformation.

From table 11 the results of the predicted nickel-based superalloys were found to be in cohesion of the state of the microstructure.

The results were predicted correctly for microstructural length-scales between tested ranges of 1 μm to 5 μm, as well as the presence of random noise in images such as text.

5.7. Software availability

A web-based application of the model was deployed using Voila (‘https://voila.readthedocs.io/’), an open-source framework used to create dashboard applications, and Heroku (‘https://devcenter.heroku.com/’) a cloud-based platform service. The URL of the deployed app is ‘https://strain-classification.herokuapp.com/’.

6. Conclusion

This paper investigates the feasibility of using deep learning approaches to classify nickel-based superalloys. It involved using pre-trained models on phase-field simulation data to train various models with different hyper-parameter values. The optimized hyper-parameter values were then applied to the second training phase with SEM images of nickel-based superalloy. The fine-tuning procedure was carried out to help alleviate the shortage of experimental data when training the model.

ResNet34 architecture was found to be the most appropriate for the strain classification task. The model achieved an accuracy of 97.74% without data augmentation and 97.63% with data augmentation with the phase-field simulation data. After fine-tuning, it achieved an accuracy of 100% on the experimental images, as opposed to an accuracy of 98.86% without fine-tuning.

The study shows that simulation datasets can be used in training models in a similar domain to produce accurate and efficient models by applying the fine-tuned parameters to the experimental dataset. The model also proved to be independent of image scale size ranging between 1 μm to 5 μm were found to give accurate results.

Acknowledgments

The authors acknowledge funding from the German Science Foundation (DFG) in the framework of the Collaborative Research Centre/Transregio 103 (Projects A1, C5, and T2).

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.
ORCID iDs

Abdulmonem Obaied © https://orcid.org/0000-0002-9641-1749
Ingo Steinbach © https://orcid.org/0000-0002-2834-1160
Irina Roslyakova © https://orcid.org/0000-0003-4329-6502

References

[1] Kim C 2012 Creep damage characterization of Ni-based superalloy by acoustic nonlinearity Prog. Nat. Sci.: Mater. Int. 22 303–10
[2] Kvakilova M, Dvorak J, Kral P, Hrbacek K and Sklenicka V 2019 Creep behaviour and life assessment of a cast nickel—base superalloy MAR—M247 High Temp. Mater. Processes 38 590–600
[3] Rösler J, Näth O, Jäger S, Schmitz F and Mukherji D 2005 Fabrication of nanoporous Ni-based superalloy membranes Acta Mater. 53 1397–406
[4] Yu Z, Wang X, Yang F, Yue Z and Li J C M 2020 Review of γ’ rafting behavior in nickel-based superalloys: crystal plasticity and phase-field simulation Crystals 10 1095
[5] Wu X et al 2020 Unveiling the Re effect in Ni-based single crystal superalloys Nat. Commun. 11 389
[6] Mali H S and Unune D R 2017 Machinability of nickel-based superalloys: an overview Available at https://sciedirect.com/science/article/pii/B9780128035818098179
[7] Long H et al 2020 A comparative study of rafting mechanisms of Ni-based single crystal superalloys Mater. Des. 196 109097
[8] Xia W, Zhao X, Yue L and Zhang Z 2020 A review of composition evolution in Ni-based single crystal superalloys J. Mater. Sci. Technol. 44 76–95
[9] Ahmed M, Horst O M, Obaied A, Steinbach I and Roslyakova I 2021 Automated image analysis for quantification of materials microstructure evolution Modelling Simul. Mater. Sci. Eng. 29 055012
[10] Ratel N, Bruno G, Bastie P and Mori T 2006 Plastic strain-induced rafting of precipitates in Ni superalloys: elasticity analysis Acta Mater. 54 5087–93
[11] DeCost B L and Holm E A 2015 A computer vision approach for automated analysis and classification of microstructural image data Comput. Mater. Sci. 110 126–33
[12] O’ Mahony N, Campbell S, Carvalho A, Harapanahalli S, Hernandez G V, Krpalkova L, Riordan D and Walsh J 2019 Deep learning vs traditional computer vision (arXiv:abs/1910.13796)
[13] Schmidt J, Marques M R G, Botti S and Marques M A L 2019 Recent advances and applications of machine learning in solid-state materials science npj Comput. Mater. 5 83
[14] LeCun Y, Bengio Y and Hinton G 2015 Deep learning Nature 521 436–44
[15] Lecun Y, Bottou L, Bengio Y and Haffner P 1998 Gradient-based learning applied to document recognition Proc. IEEE 86 2278–324
[16] Krizhevsky A, Sutskever I and Hinton G E 2017 Imagenet classification with deep convolutional neural networks Commun. ACM 60 84–90
[17] Zeiler M D and Fergus R 2013 Visualizing and understanding convolutional networks (arXiv:1311.2901)
[18] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2014 Going deeper with convolutions (arXiv:1409.4842)
[19] Simonyan K and Zisserman A 2015 Very deep convolutional networks for large-scale image recognition (arXiv:1409.1556)
[20] He K, Zhang X, Ren S and Sun J 2015 Deep residual learning for image recognition (arXiv:1512.03385)
[21] Azimi S M, Britz D, Engstler M, Fritz M and Mücklich F 2018 Advanced steel microstructural classification by deep learning methods Sci. Rep. 8 2128
[22] Chowdhury A, Kautz E, Yener B and Lewis D 2016 Image driven machine learning methods for microstructure recognition Comput. Mater. Sci. 123 176–87
[23] Iglesias J C Á, Santos R B M and Paciornik S 2019 Deep learning discrimination of quartz and resin in optical microscopy images of minerals Miner. Eng. 138 79–85
[24] Feng S, Zhou H and Dong H 2018 Using deep neural network with small dataset to predict material defects Mater. Des. 162 300–10
[25] Lin Y C, Li J, Chen M-S and Chen X-M 2016 A comparative study on phenomenon and deep belief network models for hot deformation behavior of an Al–Zn–Mg–Cu alloy Appl. Phys. A 123 68
[26] Lin Y C, Li J, Chen M-S and Liang Y-Y and Liang Y-J 2018 Convolutional neural networks for hot deformation behavior of a Ni-based superalloy Neural. Comput. Appl. 29 1015–23
[27] Chen D-D, Lin Y C and Wu F 2019 A design framework for optimizing forming processing parameters based on matrix cellular automat and neural network-based model predictive control methods Appl. Math. Modelling 76 918–37
[28] Lin Y C, Huang J, Li H-B and Chen D-D 2018 Phase transformation and constitutive models of a hot compressed TC18 titanium alloy in the α+β regime Vacuum 157 83–91
[29] Yamashita R, Nishio M, Do R K and Inoue M 2018 Convolutional neural networks: an overview and application in radiology Insights Imaging 9 611–29
[30] Scherer D, Müller A and Behnke S 2010 Evaluation of pooling operations in convolutional architectures for object recognition Artificial Neural Networks—ICANN 2010 (Lecture Notes in Computer Science) ed K Diamantaras, W Duch and L S Iliadis (Springer) pp 92–101
[31] Steinbach I, Ahmed M, Ali M A and Roslyakova I 2019 3D phase-field simulations to machine-learn 3D features from 2D microstructure Materials Research Meeting (Yokohama, Japan) (invited talk)
[32] Ali M A, Görler J V and Steinbach I 2020 Role of coherency loss on rafting behavior of Ni-based superalloys Comput. Mater. Sci. 171 109279
[33] Ali M A, Amin W, Shchyglo O and Steinbach I 2020 45-degree rafting in Ni-based superalloys: a combined phase-field and strain gradient crystal plasticity study Int. J. Plast. 128 102659
[34] Ali M A, López-Galilea I, Gao S, Ruttert B, Amin W, Shchyglo O, Hartmaier A, Theisen W and Steinbach I 2020 Effect of γ′ precipitate size on hardness and creep properties of Ni-base single crystal superalloys: experiment and simulation Materialia 12 100692
[35] Horst O M 2017 Zum Einfluss von heißisostatisch Pressen auf das Kriechverhalten einkristalliner Superlegierungen Master’s Thesis Ruhr-University Bochum
[36] Horst O, Ruttert B, Bürger D, Heep L, Wang H, Dlouhý A, Theisen W and Eggeler G 2019 On the rejuvenation of crept Ni-Base single crystal superalloys (SX) by hot isostatic pressing (HIP) Mater. Sci. Eng. A 758 202–14
[37] Shorten C and Khoshgoftaar T M 2019 A survey on image data augmentation for deep learning J. Big Data 6 60
[38] Van Rossum G and Drake F L 2009 Python 3 Reference Manual (Scotts Valley, CA: CreateSpace)
[39] Paszke A et al 2019 PyTorch: an imperative style, high-performance deep learning library Advances in Neural Information Processing Systems vol 32 (Curran Associates, Inc.) pp 8024–35 Available at http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[40] Howard J et al 2018 Fastai https://github.com/fastai/fastai
[41] Howard J and Gugger S 2020 Fastai: a layered API for deep learning Information 11 108
[42] Trampert P, Rubinstein D, Boughorbel F, Schlinkmann C, Luschkova M, Slusallek P, Dahmen T and Sandfeld S 2021 Deep neural networks for analysis of microscopy images—synthetic data generation and adaptive sampling Crystals 11 258
[43] Rakocz N et al 2021 Automated identification of clinical features from sparsely annotated three-dimensional medical imaging npj Digit. Med. 4 44
[44] Kumar S, Mishra S and Singh S K 2020 Deep transfer learning-based Covid-19 prediction using chest x-rays medRxiv preprint https://doi.org/10.1101/2020.05.12.20099937
[45] Deng J, Dong W, Socher R, Li L-J, Li K and Fei-Fei L 2009 ImageNet: a large-scale hierarchical image database 2009 IEEE Conf. Computer Vision and Pattern Recognition
[46] Wilson D R and Martinez T R 2003 The general inefficiency of batch training for gradient descent Neur. Netw. 16 1429–51
[47] Smith L N 2017 Cyclical learning rates for training neural networks (arXiv:1506.01186)
[48] Brownlee J 2019 Understand the impact of learning rate on neural network performance Available at https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/
[49] Rojas R 1996 The backpropagation algorithm Neural Netw. 149–82
[50] LeNail A 2019 NN-SVG: publication-ready neural network architecture schematics J. Open Source Softw. 4 747
[51] Roslyakova I, Zomorodpoosh S, Ali M A, Obaied A and Steinbach I 2020 Artificial materials intelligence to accelerate discovery of novel superalloys 17th Discussion Meeting on Thermodynamics of Alloys (TOFA) (Germany, Bad Staffelstein 28 September–2 October 2020) (invited talk)
[52] Desmorat R, Mattiello A and Cormier J 2017 A tensorial thermodynamic framework to account for the $\gamma''$ rafting in nickel-based single crystal superalloys Int. J. Plast. 95 43–81
[53] Epishin A, Link T, Brückner U and Portella P D 2001 Kinetics of the topological inversion of the $\gamma/\gamma'$-microstructure during creep of a nickel-based superalloy Acta Mater. 49 4017–23