applied sciences

MDPI

Article

A Deep Learning Approach to Semantic Segmentation of Steel Microstructures

Jorge Muñoz-Rodenas [1], Francisco García-Sevilla [1,2,*], Valentín Miguel-Eguía [1,2,*], Juana Coello-Sobrino [1,2] and Alberto Martínez-Martínez [2]

[1] High Technical School of Industrial Engineering of Albacete, Castilla-La Mancha University, 02006 Albacete, Spain; jjmr25@educastillalamancha.es (J.M.-R.); juana.coello@uclm.es (J.C.-S.)
[2] Materials Science and Engineering Laboratory, Regional Development Institute, Castilla-La Mancha University, 02006 Albacete, Spain; alberto.martinez@uclm.es
* Correspondence: francisco.garcia@uclm.es (F.G.-S.); valentin.miguel@uclm.es (V.M.-E.)

Featured Application: A segmentation tool for microconstituents recognition in steel optical micrographs.

Abstract: The utilization of convolutional neural networks (CNNs) for semantic segmentation has proven to be successful in various applications, such as autonomous vehicle environment analysis, medical imaging, and satellite imagery. In this study, we investigate the application of different segmentation networks, including Deeplabv3+, U-Net, and SegNet, each recognized for their effectiveness in semantic segmentation tasks. Additionally, in the case of Deeplabv3+, we leverage the use of pre-trained ResNet50, ResNet18 and MobileNetv2 as feature extractors for a comprehensive analysis of steel microstructures. Our specific focus is on distinguishing perlite and ferrite phases in micrographs of low-carbon steel specimens subjected to annealing heat treatment. The micrographs obtained using an optical microscope are manually segmented. Preprocessing techniques are then applied to create a dataset for building a supervised learning model. In the results section, we discuss in detail the performance of the obtained models and the metrics used. The models achieve a remarkable 95% to 98% accuracy in correctly labeling pixels for each phase. This underscores the effectiveness of our approach in differentiating perlite and ferrite phases within steel microstructures.

Keywords: deep learning; segmentation; low-carbon steels; optical microstructure

Citation: Muñoz-Rodenas, J.; García-Sevilla, F.; Miguel-Eguía, V.; Coello-Sobrino, J.; Martínez-Martínez, A. A Deep Learning Approach to Semantic Segmentation of Steel Microstructures. *Appl. Sci.* **2024**, *14*, 2297. https://doi.org/10.3390/app14062297

Academic Editors: Glenn Hawe and Aidan Meade

Received: 20 February 2024
Revised: 5 March 2024
Accepted: 6 March 2024
Published: 8 March 2024

1. Introduction

The recognition of the microconstituents of a steel micrograph is a complicated task and only within the reach of highly qualified personnel with broad experience in the field of materials science. Manual identification of steel phases can be a tedious and error-prone task; therefore, machine learning (ML) models have emerged as valuable complements to the traditional visual inspection methods employed by metallurgists. In recent years, many studies have addressed the challenge of developing artificial intelligence techniques that enable computers to handle complex tasks, such as microstructure identification [1–3] and the inference of properties through these identification techniques utilizing ML has been investigated [4–7], yielding promising advancements. Nevertheless, given the complexity involved in microstructure identification, particularly within steel micrographs, the adoption of advanced techniques becomes necessary. For the realization of an effective image segmentation in the context of steel microstructures, a powerful tool, such as a deep neural network, is required.

In previous work [8], it was determined that, for the categorization of steel microstructures, convolutional neural networks exhibit a notable superiority over classical machine learning algorithms. The present study constitutes a continuation of the exploration of deep learning techniques within the domain of optical micrographs of carbon steels, with a focus on segmentation algorithms. These networks allow us to establish a labeling of each pixel according to the phase of the microconstituent to be classified by means of supervised learning methods.

Recent advancements in the field of steel microstructure identification using segmentation techniques can be found. The following are discussed to provide background and context for the present article.

Luengo et al. [9] present a comprehensive overview of AI techniques for metallographic image segmentation, utilizing two distinct datasets: The Ultra-High Carbon Steel Micrograph Database (UHCSM) and the Metallography Dataset from Additive Manufacturing of Steels (MetalDAM). The paper contributes significantly by introducing the novel dataset, MetalDAM, available at https://dasci.es/transferencia/open-data/metal-dam/, accessed on 23 June 2023, providing an updated taxonomy of segmentation methods and exploring various deep learning-based ensemble strategies. Ensemble models exhibit superior performance in segmentation, achieving an Intersection over Union (IoU) metric of 76.71 for the UHCS dataset and 67.77 for the MetalDAM dataset. However, the performance achieved in both datasets is low. The authors conclude that microstructure segmentation faces limitations due to the insufficient availability of large datasets, the absence of pre-trained models tailored to this domain, and the notable challenges related to generalization errors in machine learning methods.

Bulgarevich et al. [10] address the challenge of segmenting optical images of microstructures using a supervised machine learning approach. They employ the Random Forest (RF) algorithm along with image processing and segmentation protocols, including Euclidean distance conversion and structure tensor extraction, for accurate image analysis. This research recognizes the RF algorithm as a highly versatile method for segmenting various microstructures, such as ferrite, pearlite, bainite, martensite, and martensite–austenite, within steel microstructures. The results demonstrate that the segmentation quality achieved is practical and allows meaningful statistics on the volume fraction of each phase to be obtained.

Bachmann et al. [11] present an exhaustive approach for detecting prior austenite grains (PAGs) in Nital-etched micrographs of bainitic and martensitic steels. The study utilizes a correlative microscopy technique, combining a light optical microscope (LOM), a scanning electron microscope (SEM), and electron backscatter diffraction (EBSD). The detection of PAGs is accomplished through semantic segmentation using advanced deep learning (DL) methods, specifically U-NET in conjunction with DenseNet, applied to LOM images.

To ensure effective model evaluation, the authors emphasize the critical importance of accurately measuring grain sizes in the metallurgical structure of the material. Their experiments reveal an IoU of around 70%, indicating potential discrepancies between metric values and visual perception of model quality. Recognizing the limitations of traditional metrics like IoU and pixel accuracy, particularly in the context of grain size measurement within segmentation tasks, they propose a novel approach. To address this, they introduce a method for quantifying grain size distribution from segmentation maps, calculating the mean, median, and standard deviation. By binning detected grains into intervals of a specific width (500 μm^2) and calculating probability density, they accurately assess segmentation quality compared with values of the ground truth and identify potential errors in grain size determination. The results show a mean error of 6.1% in average grain size, underscoring the high quality of the DL model.

Han et al. [12] introduced a segmentation method (CES) based on the extraction of center–environment features tailored for small material image samples. The proposed method is applied to several datasets that include carbon steels, titanium alloy, wood, and cross-sectional morphology of Pt-Al and WC-Co coating image data. Expert annotators are engaged in the process, drawing region-specific curves based on their domain knowledge. Additionally, the method takes advantage of several machine learning algorithms to achieve highly accurate segmentation. Notably, the results of the study indicate that the Gradient Boosting Decision Tree (GBDT) outperforms other methods in this context.

Additionally, a comparison is made with segmentation methods based on deep learning networks such as SegNet, PSPNet, and UNet++, which are found to be 10% higher in IoU and mean IoU metrics compared to the methodology used by the authors. This difference is attributed to the significantly fewer pixels annotated to create the masks using CES compared to deep learning methods. While the proposed method is commendable for its innovative approach and reduced annotation cost, it falls short in achieving comparable segmentation accuracy to deep learning algorithms. The observed 10% disparity in results highlights the limitations of this method, suggesting that a balance between annotation efficiency and segmentation performance has yet to be fully realized.

Kim et al. [13] displayed the segmentation of a low-carbon steel microstructure without the need for labeled images, employing a deep learning approach. Specifically, a convolutional neural network combined with the Simple Linear Iterative Clustering (SLIC) superpixel algorithm. By leveraging a diverse range of microstructure optical images containing ferrite, pearlite, bainite, and martensite, the model effectively distinguished and delineated regions corresponding to each constituent phase.

Breumier et al. [14] trained a U-Net model to perform the segmentation of bainite, ferrite and martensite on EBSD maps using the kernel average misorientation and the pattern quality index as input. The model can differentiate the three constituents with a 92% mean accuracy in the test results.

Chaurasia et al. [15] proposed a versatile approach for classifying multiphase steels. It involves generating 3D polycrystalline microstructure templates using the Johnson–Mehl–Avrami–Kolmogorov (JMAK) kinetic model, creating realistic single-phase microstructures through nucleation and growth concepts. Cropped images of pearlite and ferrite are strategically placed on these templates to synthesize accurately labeled ferrite–pearlite microstructures. Subsequently, a deep learning architecture, UNET, is trained using synthetic microstructures and tested on real microstructures. The results, compared with manually annotated microstructures, demonstrate a prominent level of agreement, reaching an accuracy of about 98%.

Liu et al. [16] conducted a study that focuses on recognizing the microconstituents of ferrite and pearlite and making predictions of their mechanical properties. For this purpose, they elaborate a residual U-shaped network based on ResNet32 to identify grain boundaries and their size, obtaining better segmentation results than the conventional neural network FCN-8s, reaching over 93% in frequency weighted intersection over union (FWIoU).

Azimi et al. [17] utilized fully convolutional networks (FCNs) along with a max-voting scheme for the classification of martensite, bainite, pearlite, and ferrite phases in low-carbon steels, achieving a classification accuracy of 93.94%.

Recently, works similar to the research in this paper have been published, such as Ostormujof et al. [18] that accomplished the successful classification of ferrite–martensite dual-phase steel microstructures through the implementation of the U-Net model and achieved pixel-wise accuracies of around 98%, as well as Xie et al. [19], who provided a comparison with different segmentation architectures for steel micrographs like DeepLabv3+, Enet, Unet, and PSPnet. They propose a new semantic network based on the improvement of a fully convolutional network (FCN) with the atrous spatial pyramid pooling (ASPP) technique for feature extraction, surpassing the previous ones according to the metric Intersection over Union (IoU), achieving a performance of up to 80.43%. In our specific study, we employed LOM images as opposed to the SEM images used in the referenced article. This choice might introduce differences in the characteristics and features of the micrographs, potentially impacting the performance of segmentation algorithms. It is worth noting that the selection of imaging modalities can influence the choice of segmentation techniques and their effectiveness in each context. Ma et al. [20] conducted training on two datasets comprising images of steel alloys, one consisting of carbide and the other predominantly of ferrite microconstituents. They employed PSPNet and DeepLabv3+ with ResNet18 segmentation networks. The authors proposed enhancing the receptive field of the convolutional neural network (CNN) to improve contextual perception of images without altering the network architecture. This was achieved by scaling the original image size to 0.5 times during image loading. Additionally, the authors established an automated quantitative analysis of the microstructures using OpenCV software after segmentation, extracting morphological information from classified pixels to obtain the average carbide radius and the number of carbides. The results, evaluated on original large-size images, yielded a mean Intersection over Union (mIoU) score of approximately 80%.

Additionally, Bihani et al. [21] present, in this case in the context of mudrock SEM images, a method for filtering and segmentation using deep learning to identify pore and grain features named MudrockNet, which is based on DeepLab-v3+. The predictions for the test data obtain a mean IoU of 0.6663 for silt grains, 0.7797 for clay grains, and 0.6751 for pores.

Automated phase identification in steel microstructures is a rapidly evolving field. While previous studies have addressed segmentation challenges with varying degrees of success, several issues remain unresolved, including the application of segmentation to low-magnification optical images and the scarcity of dedicated steel microstructure image databases. To address these shortcomings, this research delves into the exploration of optimal architectures for this problem, specifically targeting the development of a robust segmentation model capable of automatically identifying pearlite and ferrite phases in annealed steel microstructures, which have a major influence on the properties and behavior of annealed steels.

It can be concluded that numerous studies have explored the segmentation of steel microstructures, generating segmentation models created from ad hoc networks with varying degrees of success. Nevertheless, most experiments are conducted using data obtained from scanning electron microscopy (SEM) images, rendering them unsuitable for samples produced with optical technology. This work aims to delve deeper into obtaining segmentation models for the identification of pearlite and ferrite in images coming from optical microscopy. The Deeplabv3+ and U-Net architectures will be employed for the segmentation of LOM steel microstructure images. Leveraging convolutional neural networks, these architectures have demonstrated effectiveness in image segmentation across various domains.

The methodology employed in this study integrates ImageJ with trainable Weka segmentation, Random Forest classifier training, and data augmentation to prepare a diverse dataset for the subsequent creation and training of U-Net, SegNet and DeepLabV3+ segmentation models for steel micrograph analysis. In the following sections, we will delve into the methodology and analyze the results and discussions.

2. Materials and Methods

2.1. Steel Specimens and LOM Images

The experimental procedures involved the utilization of three steel samples that underwent annealing treatment to produce ferrite and pearlite microstructures, with their respective chemical compositions detailed in Table 1. Metallographic sample preparation was conducted by grinding and polishing according to the typical procedure used for optical microscopy and were etched with Nital-1-(alcoholic nitric acid at 1%) for 30 s, permitting observation of the grain boundaries and microstructures to be distinguished.

For the development of segmentation models, a dataset comprising 34 steel micrography images, each with a resolution of 2080 × 1542 pixels, was compiled. The selection of these images aimed to provide a comprehensive representation of the diverse microstructural features inherent in various steel samples.

Table 1. Chemical composition (weight %) of low-carbon steel samples according to ISO 683-1:2019 standards.

Steel	C	Si	Mn	P	S	Cr	Mo	Ni	Cu
C45E	0.45	0.25	0.65	0.025	0.035	0.40	0.10	0.40	0.30

As seen in Figure 1a,b, once the steel undergoes an annealing heat treatment, a crystalline structure is obtained, revealing two distinctive zones. One zone is characterized by ferrite, appearing as a whitish matrix, while the other zone appears darker with a lamellar constituent, indicating the presence of pearlite. The normalizing heat treatment results in a similar microstructure, albeit with finer constituents, as shown in Figure 1c. Figure 2 provides a detailed depiction of these constituents. As observed in Figure 2b, the pearlite consists of alternating fine bands of ferrite and cementite, maintaining a dark aspect, as mentioned earlier.

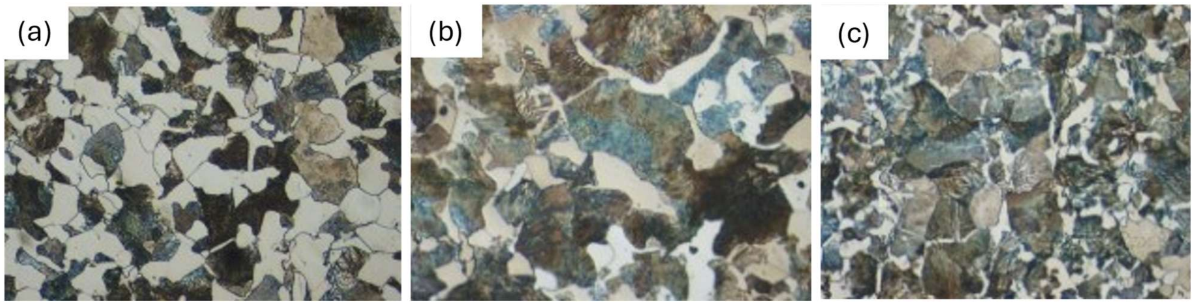


Figure 1. Samples of C45E steel in an annealing state, (**a**,**b**) and normalizing; (**c**) reagent, Nital-1.

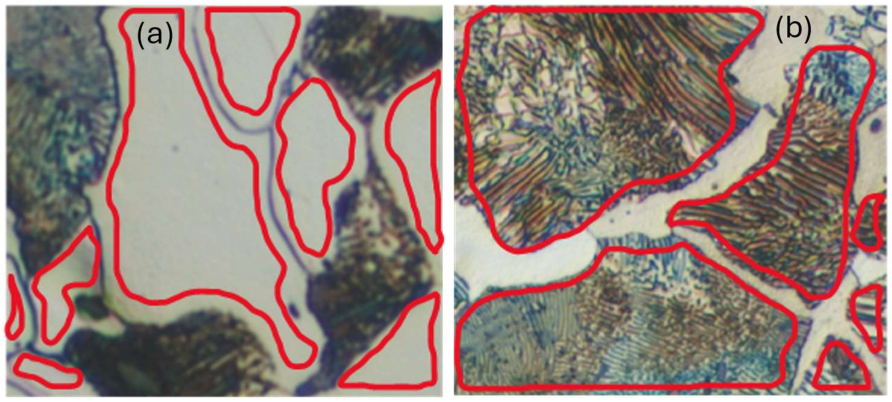


Figure 2. Microconstituents corresponding to the C45E Steel (reagent Nital-1). The red contours correspond to ferrite (**a**) and pearlite (**b**) areas.

2.2. Image Preprocessing

In the preprocessing stage of the segmentation deep learning experiment carried out in this work, a comprehensive approach was implemented to enhance the quality and diversity of the dataset. This involved the initial creation of masks using specialized software, followed by a thorough data augmentation process. ImageJ, with its trainable Weka segmentation plugin, was utilized for the creation of masks [22,23]. This allowed for the creation of masks, outlining specific regions of interest within the steel microstructure images. Manual annotations made by the authors guided the algorithm in learning the features necessary for accurate segmentation. The annotations of the pearlite areas have been manually performed on two of the original images for each sample. Subsequently, the trainable Weka segmentation option has been applied to the rest of the images to automate the generation of masks since manual mask generation is a time-consuming process and prone to errors. Thus, by using the ImageJ segmentation assistant, the quality of the masks was improved, and the processing time was reduced. Nevertheless, the authors reviewed each generated mask, making adjustments to images containing any errors.

The trainable classifier employed for mask creation was based on the Random Forest algorithm. Configured with 200 decision trees, this algorithm demonstrated robustness in handling the complexity of steel micrography images. The training process involved feeding the algorithm with the manually annotated masks, allowing it to learn and generalize patterns within the dataset. Following the initial mask creation and classifier training, a data augmentation step was introduced to enhance the dataset's diversity. This involved applying various transformations such as rotation, scaling, and flipping to the original 34 steel micrography images. The augmented dataset served to increase the model's ability to generalize across a broader range of microstructural variations.

Each original image captured by the optical microscope has a resolution of 2080 × 1542 pixels. For the execution of the experiments, we have chosen to use images of 224 × 224 pixels. This choice is based on various practical and efficiency considerations. Smaller images demand fewer computational resources for both training and inference. The utilization of 224 × 224 images enables the model to execute more rapidly. Furthermore, for the transfer learning from pretrained models utilized in the experiments, such as ResNet50, ResNet18, or MobileNetV2, these models are often trained on massive datasets with specific image sizes. Employing the same image size during both training and inference eases the transfer of knowledge from pretrained models, as the initial layers are tailored to that size. It is important to note that although 224 × 224 pixels is a commonly used size, it is not a strict constraint. The image size can be adjusted to conduct experiments with a different set of images, but it might be necessary to adjust other model parameters and, in some cases, retrain the model to accommodate the new input size.

For data augmentation, each original image and mask were cropped into 54 images of size 224 × 224 pixels. Subsequently, rotations of 90°, 180°, and 270° were applied to the cropped images, resulting in 216 images for each original image. This process yielded a final dataset of 7344 images. These images were distributed randomly, with 70% allocated for training data, 20% for validation data, and 10% for the test data.

Taking into consideration the information provided before, an example of the result of the cropping and rotating images can be appreciated in Figure 3. The masking process intended to isolate the ferrite areas contained in the images can also be observed.

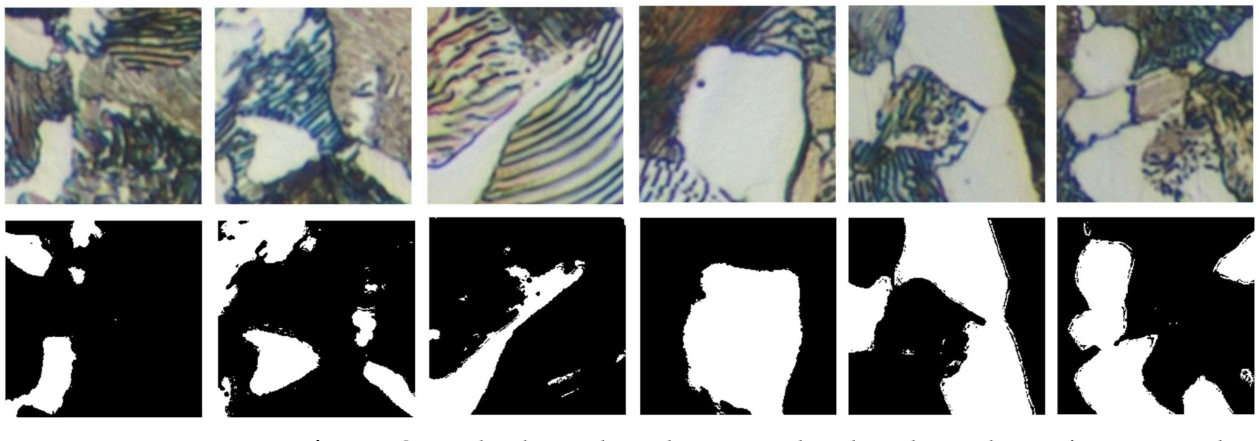


Figure 3. Cropped and rotated sample images and masks with a resolution of 224 × 224 pixels.

After the preprocessing stage was completed, the model creation phase was initiated. This involved training various segmentation models to identify important features in the preprocessed dataset. Using the enriched dataset, different model setups and methods were experimented with. The aim was to determine which approach worked best for accurately outlining the steel microstructure images. In the following section, the training process details and metrics are described.

2.3. Segmentation Model Training

In executing the experiments, various segmentation networks were employed to establish a comparative analysis and identify the most suitable one for the context of microstructures in steels subjected to an annealing heat treatment. The segmentation networks utilized include U-Net [24], SegNet [25], and DeepLabV3+ [26]. Diverse pre-trained backbones, such as ResNet18, ResNet50, and MobileNetV2, were employed for the latter.

The same algorithm has been applied to all networks. Initially, each model undergoes training using the selected images for training and validation. Once the model is generated, it is applied to the test images, subsequently obtaining various metrics [27] that facilitate result analysis. In Appendix A, comprehensive details regarding each layer within the architectures of the segmentation networks utilized are presented in tabular form. The description of the networks employed in the experiments is provided next.

2.3.1. U-Net

U-Net is commonly used in the context of semantic image segmentation, and its effectiveness in capturing both global context and fine details makes it particularly well-suited for tasks such as medical image segmentation and satellite image analysis, and it is also employed for the segmentation of materials microstructures [28,29]. U-Net is characterized by a U-shaped architecture with an encoder–decoder structure and skip connections. The encoder, on the left side of the U, consists of down-sampling layers that capture hierarchical features from the input image. The decoder, on the right side, involves up-sampling layers and skip connections that preserve high-resolution details and aid in precise localization. Skip connections connect corresponding encoder and decoder stages, facilitating the retention of spatial information. The bottleneck at the base of the U combines abstract features from the encoder with detailed spatial information from the decoder.

In the conducted experiments with U-Net, the bias term of all convolutional layers is initialized to zero. Additionally, the convolution layer weights in the encoder and decoder subnetworks are initialized using the ‘He’ weight initialization method [30]. The encoder–decoder has a depth of 3, resulting in a U-Net comprising 46 layers with 48 connections. The most relevant hyperparameters configured for training include the Adam optimizer, a learning rate of 0.001, L2 regularization, and a maximum number of epochs set to 2. Experiments were conducted by increasing the number of epochs, yet substantial improvements were not achieved; instead, there was an increase in computational time. The loss layer utilizes cross-entropy loss to quantify the disparity between the predicted values and their corresponding actual data. The formula is expressed as follows in Equation (1).

$$\mathrm{loss} = -\frac{1}{\mathrm{N}}\sum_{\mathrm{n}=1}^{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{K}} \mathrm{w_i}\,\mathrm{t_{ni}}\ln \mathrm{y_{ni}} \tag{1}$$

Here, N represents the number of samples, K is the number of classes, w_i denotes the weight for class i, t_{ni} is the indicator of whether the nth sample belongs to the ith class, and y_{ni} represents the output for sample n for class i.

2.3.2. SegNet

SegNet [31] is a convolutional neural network architecture tailored for semantic image segmentation. Its distinctive features include a conventional encoder–decoder structure, where the encoder captures hierarchical features, and the decoder reconstructs the segmented output through up-sampling layers. Notably, SegNet utilizes max-pooling indices from the encoder during decoding to recover spatial information lost during down-sampling, contributing to accurate segmentation. The network leverages feature maps from the encoder for precise localization. Employing a class-specific softmax activation in the final layer enables pixel-wise classification. Although SegNet lacks skip connections between the encoder and decoder, its design, particularly the incorporation of pooling indices, makes it well-suited for tasks demanding detailed pixel-wise segmentation.

In this study, the segmentation experiments have utilized the SegNet architecture in conjunction with VGG16 [32,33]. In this context, VGG16 plays a role as a feature extractor, capturing high-level semantic information from the input images. It complements the segmentation capabilities of SegNet, contributing to an enhanced overall performance of the segmentation model.

2.3.3. DeepLabV3+

The segmentation models were built by integrating the DeepLabV3+ architecture with various pre-trained backbones, including ResNet50, ResNet18 [34], and MobileNetV2 [35]. This diverse combination harnesses the strengths of DeepLabV3+ for pixel-wise segmentation and different backbone architectures for feature extraction. The models were trained using an augmented dataset, integrating insights obtained from the Random Forest classifier.

In Figure 4, a schematic representation of the DeepLabV3+ architecture is shown. The model employs a pretrained backbone (ResNet50, ResNet18 and MobileNetv2) for feature extraction. The Atrous Spatial Pyramid Pooling (ASPP) module is employed to capture multi-scale contextual information. The subsequent decoder, featuring skip connections, refines and up-samples the features to produce a high-resolution semantic segmentation map. This architecture provides detailed pixel-wise predictions for accurate object recognition in images.

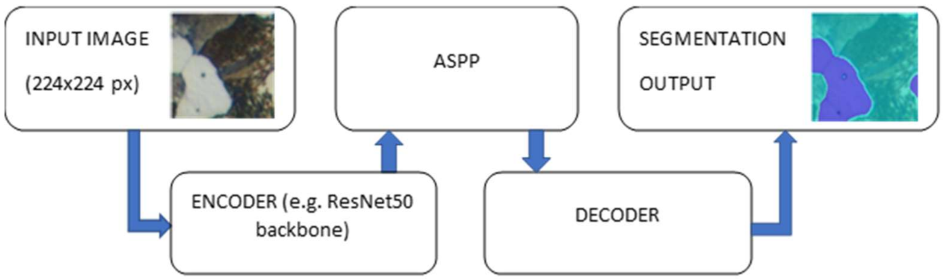


Figure 4. DeepLabV3+ and Resnet50 segmentation network architecture (adapted from [26]).

2.4. Training Parameters, Metrics and Other Details

The training process involved optimizing various parameters, including learning rates, batch sizes, and epochs. A validation set was used to monitor the model's performance and prevent overfitting.

When conducting experiments, identical training parameters were chosen to ensure a more faithful comparison of results. Adam optimizer with a learning rate of 0.001 and a maximum number of epochs set to 3 were selected. Additionally, the 'Validation Patience' parameter was set to 4 to avoid unnecessary computation. All the aforementioned information is summarized in Table 2, which compiles essential data regarding the networks for computational time considerations.

Table 2. Training parameters and network information.

Network	Optimizer	Learning Rate	Max Epochs	Batch Size	Trainable Parameters	Layers
U-Net				32	7,697,410	46
SegNet				16	29,444,166	91
DeepLabv3+ (Resnet50)	Adam	0.001	3		43,980,180	206
DeepLabv3+ (Resnet18)				32	20,607,636	100
DeepLabv3+ (MobileNet)					6,784,276	186

To evaluate the performance of the segmentation models, various metrics were employed. Accuracy measures the proportion of correctly classified pixels to the total number of pixels in each class, as defined by the ground truth, and its score is calculated using Equation (2), where TP represents true positives, and FN represents false negatives. Mean Accuracy, computed as the average Accuracy of all classes across all images, provides an aggregate assessment of model performance. Global Accuracy, on the other hand, considers the ratio of correctly classified pixels, irrespective of class, to the total number of pixels.

$$\text{Accuracy score} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

Additionally, the Boundary F1 (BF) score, known as the BF Score, evaluates the alignment between predicted boundaries and true boundaries. Calculated using Equation (3), precision assesses the accuracy of the predicted boundaries, while recall gauges the model's ability to capture true boundaries. A higher BF score indicates better agreement between predicted and true boundaries. The Mean BF Score offers an aggregate measure of boundary prediction performance across all classes and images.

$$\text{BF score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

Furthermore, the Intersection over Union (IoU) score assesses the ratio of correctly classified pixels to the total number of ground truth and predicted pixels in each class. The IoU score is computed using Equation (4), where TP represents true positives, FP represents false positives, and FN represents false negatives. The Mean IoU provides an average IoU score across all classes and images, offering insights into the overall segmentation accuracy of the model.

$$\text{IoU score} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{4}$$

The trained segmentation models were evaluated on a separate test set of steel micrograph images not seen during training. The metrics used for the evaluation of the models have been previously specified.

All experiments were conducted on a robust computing system equipped with an Intel(R) Core(TM) i7-5930K CPU @ 3.50 GHz, DIMM 64 GB RAM, and an NVIDIA® GEFORCE RTX 3080 (10 GB). MATLAB® was utilized for coding and generating segmentation models, and ImageJ was employed for mask creation, guaranteeing a stable and reproducible computational environment. All codes performed for this research are available upon request.

3. Results

Five different models were trained using 5141 training images and 1469 validation images. The training results are presented in Table 3, and the training progress can be observed in Figures 5–9 for each of the models. The progression of both accuracy and the loss function is depicted.

Table 3. Training results (LR = 0.001) (bold numbers represent the maximum values).

Model	Training Accuracy (%)	Training Loss	Final Validation Accuracy (%)	Final Validation Loss	Output Network Iteration	Time Elapsed (hh:mm:ss)
U-Net	95.874	0.128	96.553	0.095	320	00:06:50
DeepLabv3+resn50	97.185	0.068	97.485	0.060	480	00:08:14
DeepLabv3+resn18	97.229	0.068	97.177	0.071	480	00:04:49
DeepLabv3+mobn	**97.379**	**0.063**	**97.969**	**0.050**	480	00:07:41
SegNet	95.359	0.236	96.773	0.169	963	00:20:23

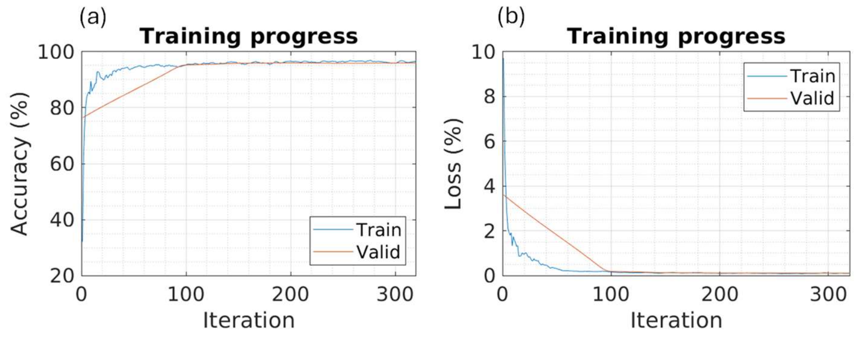


Figure 5. Training progress of U-Net: (**a**) training and validation accuracy; (**b**) training and validation loss.

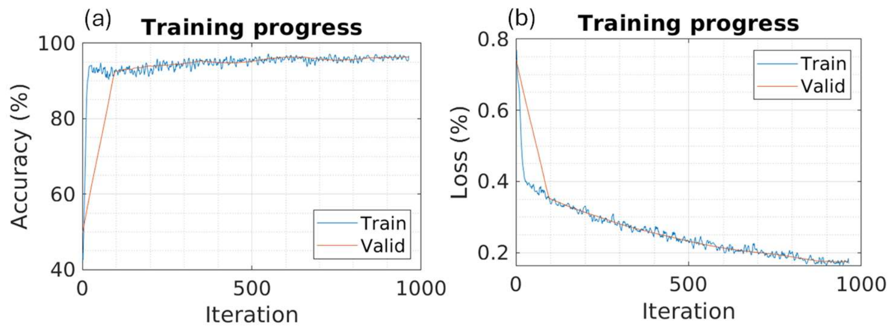


Figure 6. Training progress of SegNet: (**a**) training and validation accuracy; (**b**) training and validation loss.

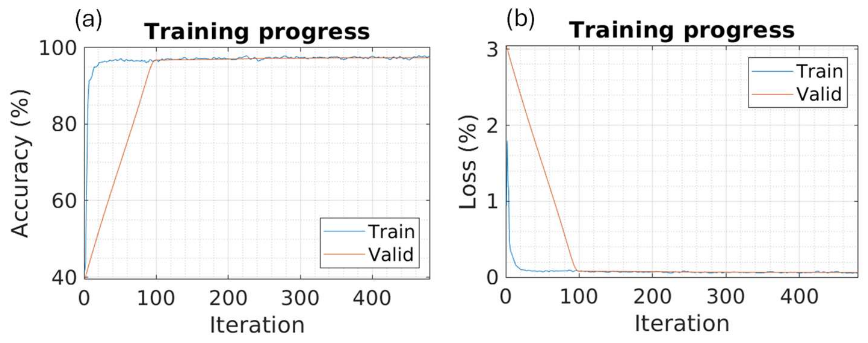


Figure 7 Training progress of DeepLabv3+/ResNet50: (**a**) training and validation accuracy; (**b**) training and validation loss.

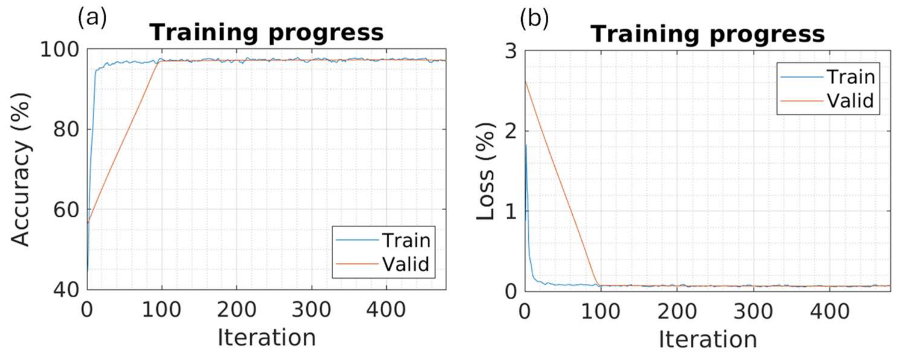


Figure 8. Training progress of DeepLabv3+/ResNet18: (**a**) training and validation accuracy; (**b**) training and validation loss.

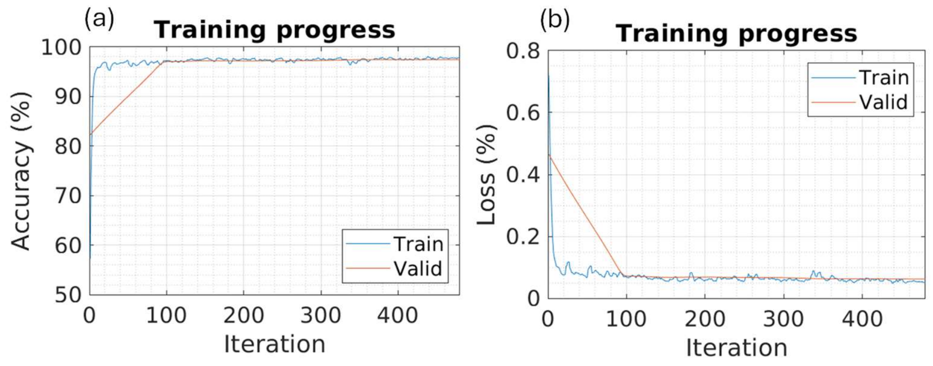


Figure 9. Training progress of DeepLabv3+/MobileNetv2: (**a**) training and validation accuracy; (**b**) training and validation loss.

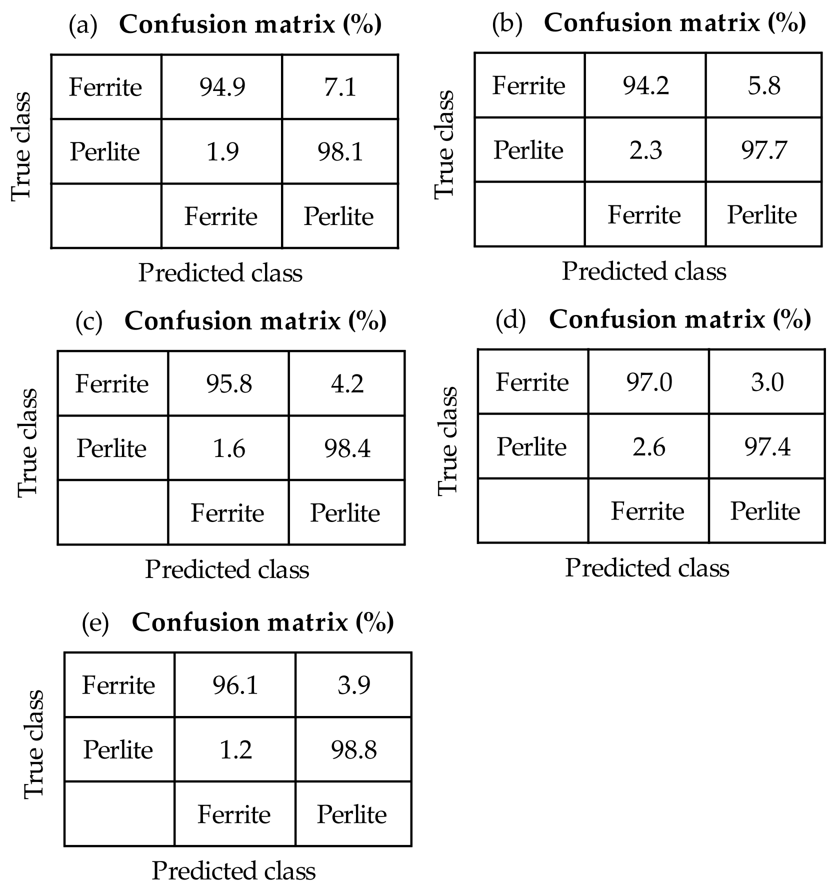
The model was then applied to 734 test images that it had not seen previously. The results of the test experiments are included in Table 4, which displays the usual metrics for segmentation problems. Additionally, the confusion matrices are shown in Figure 10. It can be inferred that the DeepLabv3+ model with MobileNetv2 achieves a performance improvement, though only slightly surpassing the other networks that also accurately solve the segmentation problem.

To visually explore the results, two test images were utilized, and each was processed by every trained model. These images are depicted in Figure 11. The segmentation performed by each model can be observed for comparison with the original sample, as well as with the mask or ground truth generated during data preprocessing before training. The objective is to distinguish between the two microconstituents: ferrite as the matrix element represented by the lighter zone in the micrograph and pearlite composed of alternating layers of cementite and ferrite. It is crucial to emphasize that the ferrite constituting the pearlite should not be segmented together with the ferrite, forming the matrix of the microstructure.

In the training phase, it can be observed that the SegNet model requires more iterations and, consequently, more computational time to achieve maximum accuracy, as depicted in Figure 6, exceeding more than twice the others. However, its final training accuracy does not differ significantly from the rest, trailing only by a couple of percentage points compared to DeepLabv3+, which yields the best results. This increased number of iterations is due to the reduction in MiniBatchSize to 16 samples for SegNet, compared to the MiniBatchSize of 32 samples used for the other networks. Notably, when employing a MiniBatchSize of 32 samples, the performance of SegNet decreases to approximately 91% to 93%, emphasizing the need to reduce the MiniBatchSize to 16 for optimal performance. Despite the longer training time associated with the reduced MiniBatchSize, SegNet's final accuracy remains competitive, showcasing its ability to achieve high performance even with a smaller batch size. As shown in Figures 7–9 achieving maximum accuracy during training requires only a few iterations for DeepLabv3+ segmentation networks. The encoder that leads to the shortest training time is ResNet18, which has the fewest layers among the three. However, MobileNetV2 exhibits slightly superior results to the other networks, achieving excellent scores in all metrics as indicated in Table 4.

Table 4. Test image metrics (bold numbers represent the maximum values).

Model	Global Accuracy	Mean Accuracy	Mean IOU	Weighted IOU	Mean BF Score
U-Net	0.9667	0.9551	0.9202	0.9359	0.8578
DeepLabv3+ResNet50	0.9757	0.9722	0.9418	0.9529	0.8798
DeepLabv3+ResNet18	0.9725	0.9717	0.9349	0.9472	0.8471
DeepLabv3+MobNetv2	**0.9802**	**0.9743**	**0.9521**	**0.9614**	**0.9149**
SegNet	0.9675	0.9596	0.9229	0.9377	0.8127



(a) **Confusion matrix (%)**

True class

Ferrite	94.9	7.1
Perlite	1.9	98.1
	Ferrite	Perlite

Predicted class

(b) **Confusion matrix (%)**

True class

Ferrite	94.2	5.8
Perlite	2.3	97.7
	Ferrite	Perlite

Predicted class

(c) **Confusion matrix (%)**

True class

Ferrite	95.8	4.2
Perlite	1.6	98.4
	Ferrite	Perlite

Predicted class

(d) **Confusion matrix (%)**

True class

Ferrite	97.0	3.0
Perlite	2.6	97.4
	Ferrite	Perlite

Predicted class

(e) **Confusion matrix (%)**

True class

Ferrite	96.1	3.9
Perlite	1.2	98.8
	Ferrite	Perlite

Predicted class

Figure 10. Confusion Matrix: (**a**) U-Net, (**b**) SegNet, (**c**) DeepLabv3+/ResNet50, (**d**) DeepLabv3+/ResNet18, and (**e**) Mobilenetv2.

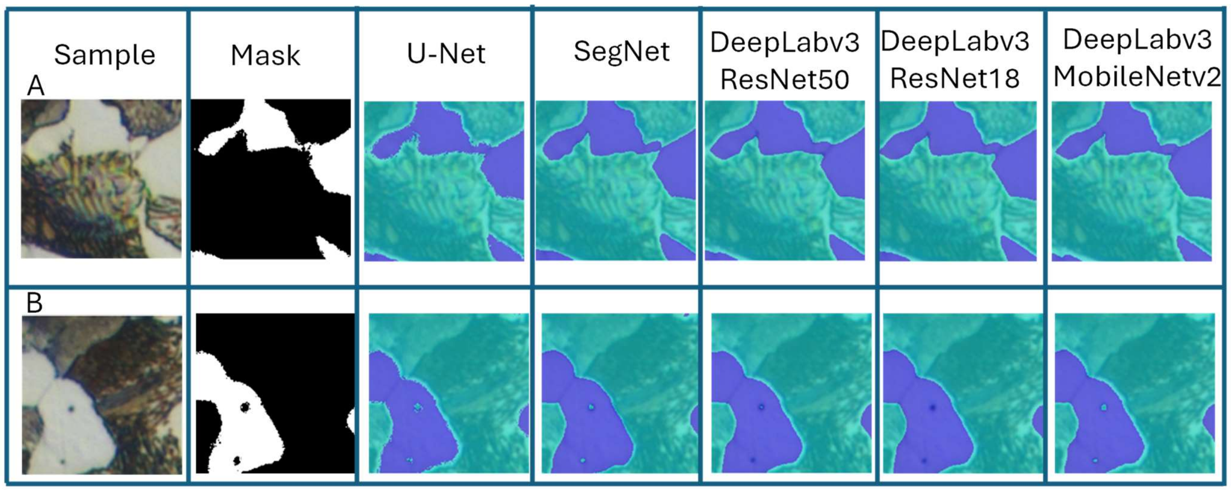


Figure 11. Segmented samples. A and B correspond to two randomly selected samples.

During the training process of the segmentation model, anomalies or irregularities that might occur in individual images are likely to diminish or be addressed as the model learns from a diverse set of images. The learning process, driven by probabilities, helps the model to generalize and effectively segment objects or regions of interest in images, even in cases where there might be variations or anomalies in the data. In this case, the model might not learn extensively about these imperfections due to their limited occurrence in the training data.

4. Discussion

Different random test samples were selected for segmentation using the obtained models. The accuracy and loss values in Figures 5–9 are obtained during training. The overall values, as shown in Table 4, are calculated based on test images that the model had not previously seen. These test values closely resemble those observed during training, indicating that no "overfitting" has occurred in any of the models.

Algorithms with lower loss rates and higher accuracy during training may demonstrate superior generalization performance on unseen data, resulting in higher final accuracy.

In Figure 11 (segmented samples), a comparison of 224 × 224 images of annealed steel is presented, highlighting the region considered as perlite in green hues and the matrix or ferrite, which appears light in the original image and violet in the segmented image. The grayscale image corresponds to the mask generated during data preprocessing. Although the results are very similar, subtle differences can be perceived. It is important to note that some masks were created manually, while the rest underwent preprocessing using a Random Forest algorithm with WEKA software. This process may have introduced errors in pixel annotation in some masks, causing the model to learn from imperfect images. As shown in microstructure A, there is an error in the bottom right part of the mask (slightly pointed area), Figure 12a, where the ferrite zone connecting with the one in the top right has not been completely obtained. This flaw is highlighted in red in Figure 12b. This error has also been transferred to the training models, which consequently failed to detect the ferrite in that zone. However, a slight improvement in the segmented area compared to the mask is noticeable. Similarly, in image B, impurities can be observed on the ferrite area (two dots on the left side), which were also transferred to the training dataset. In this case, models like DeepLabv3 with ResNet50-18 have effectively eliminated these impurities during the segmentation process.

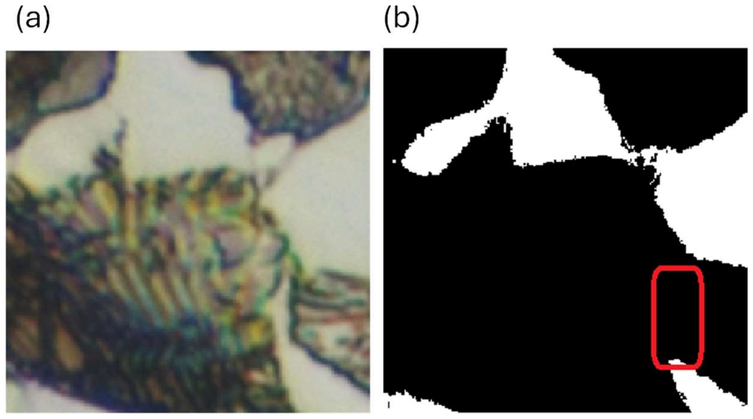


Figure 12. Detail of the error in mask production during preprocessing. (**a**) Image from test dataset; (**b**) mask. The red box indicates the lack of ferrite in the mask.

As shown in Figure 13, another test sample was selected, and errors in the identification of ferrite and perlite were marked on the corresponding mask image. The segmented images by the models demonstrate improvement over the mask created for training. We can observe that in the original image, it is difficult to appreciate the laminar structure of perlite. Although ferrite, as the matrix element of the microstructure, should be easily detected due to its more uniform and clear texture, the models encounter issues in some areas, such as the band in Figure 13b, which is indicated in the red rectangular area. Considering perlite as alternating layers of ferrite and cementite, the thickness of this bright band between two darker zones causes the models to interpret that area as perlite. The models with DeepLabv3+/MobileNetv2, shown in Figure 13f and, to some extent, U-Net, manage to enhance segmentation in that specific area.

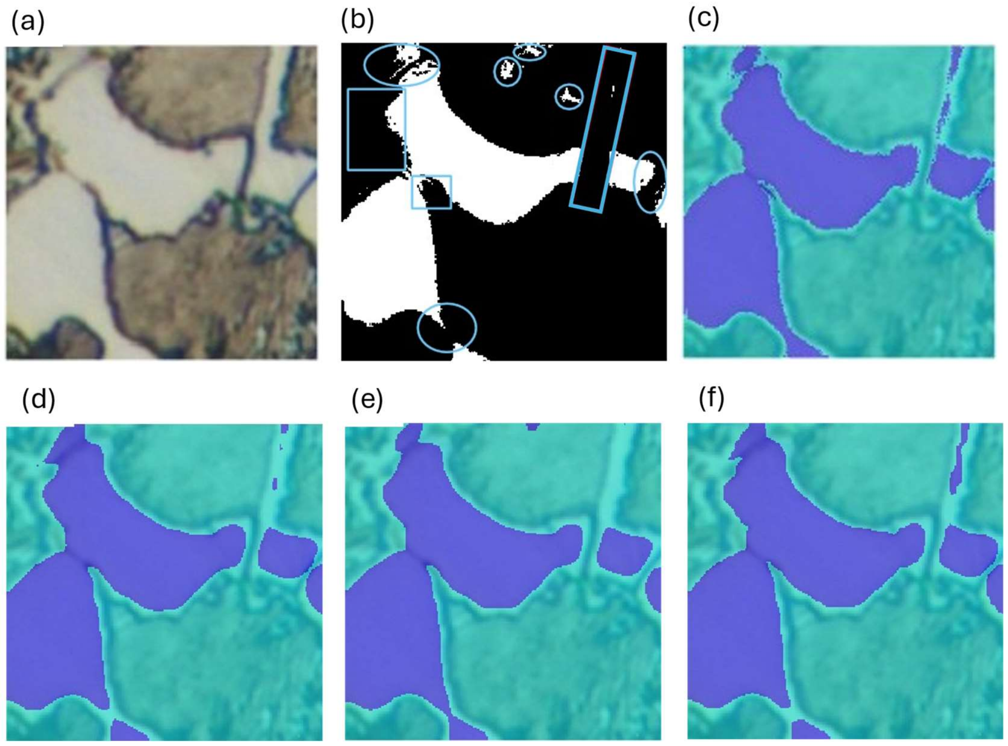


Figure 13. U-Net, (**a**) test image sample, (**b**) mask of the sample, (**c**) semantic segmentation U-NET, (**d**) DeepLabv3+/ResNet50, (**e**) DeepLabv3+/ResNet18, and (**f**) Deeplabv3/MobileNetv2.

5. Conclusions

This study investigates the identification of microconstituents, specifically ferrite and perlite, in optical metallographic images of steels using deep learning networks specialized in image segmentation problems. The work encompasses challenging tasks, particularly in obtaining and preparing the images. While other studies often concentrate on detecting various microconstituents using electron microscopy, where differences are typically more pronounced, our focus is on optical images. As the core of this study is grounded in optical images, a preliminary investigation has been undertaken on microconstituents derived from annealing heat treatments.

Segmenting distinct and clearly identifiable textures, such as perlite and ferrite, could be approached using classical algorithms with the application of conventional computer vision filters or classical machine learning techniques. However, in other studies conducted by the authors, it has been confirmed that the application of deep learning techniques to steel metallographic images enhances the metrics compared to classical machine learning algorithms. The advantage of approaching the study through deep learning is the creation of models that can be integrated into more general models in the future through transfer learning or model ensemble, thereby forming a superior structure.

Author Contributions: Conceptualization, F.G.-S. and V.M.-E.; methodology, J.M.-R., F.G.-S., J.C.-S., A.M.-M. and V.M.-E.; software, J.M.-R. and F.G.-S.; validation, J.C.-S., A.M.-M. and J.M.-R.; formal analysis, J.M.-R., F.G.-S. and V.M.-E.; investigation, J.M.-R., A.M.-M. and J.C.-S.; resources, A.M.-M., J.C.-S. and V.M.-E.; data curation, J.M.-R. and F.G.-S.; writing original draft preparation, J.M.-R.; writing review and editing, J.M.-R., F.G.-S. and V.M.-E.; visualization, F.G.-S. and V.M.-E.; supervision, F.G.-S., J.C.-S., A.M.-M. and V.M.-E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data will be made available upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. U-Net Layer information (Activation format: S—Spatial; C–Channel; B—Batch).

	Name	Type	Activations					Learnables
1	ImageInputLayer	Image Input	224	224	3	1	SSCB	0
2	Encoder-Stage-1-Conv-1	2-D Convolution	224	224	64	1	SSCB	1792
3	Encoder-Stage-1-ReLU-1	ReLU	224	224	64	1	SSCB	0
4	Encoder-Stage-1-Conv-2	2-D Convolution	224	224	64	1	SSCB	36,928
5	Encoder-Stage-1-ReLU-2	ReLU	224	224	64	1	SSCB	0
6	Encoder-Stage-1-MaxPool	2-D Max Pooling	112	112	64	1	SSCB	0
7	Encoder-Stage-2-Conv-1	2-D Convolution	112	112	128	1	SSCB	73,856
8	Encoder-Stage-2-ReLU-1	ReLU	112	112	128	1	SSCB	0
9	Encoder-Stage-2-Conv-2	2-D Convolution	112	112	128	1	SSCB	147,584
10	Encoder-Stage-2-ReLU-2	ReLU	112	112	128	1	SSCB	0
11	Encoder-Stage-2-MaxPool	2-D Max Pooling	56	56	128	1	SSCB	0
12	Encoder-Stage-3-Conv-1	2-D Convolution	56	56	256	1	SSCB	295,168
13	Encoder-Stage-3-ReLU-1	ReLU	56	56	256	1	SSCB	0
14	Encoder-Stage-3-Conv-2	2-D Convolution	56	56	256	1	SSCB	590,080

15	Encoder-Stage-3-ReLU-2	ReLU	56	56	256	1	SSCB	0
16	Encoder-Stage-3-DropOut	Dropout	56	56	256	1	SSCB	0
17	Encoder-Stage-3-MaxPool	2-D Max Pooling	28	28	256	1	SSCB	0
18	Bridge-Conv-1	2-D Convolution	28	28	512	1	SSCB	1,180,160
19	Bridge-ReLU-1	ReLU	28	28	512	1	SSCB	0
20	Bridge-Conv-2	2-D Convolution	28	28	512	1	SSCB	2,359,808
21	Bridge-ReLU-2	ReLU	28	28	512	1	SSCB	0
22	Bridge-DropOut	Dropout	28	28	512	1	SSCB	0
23	Decoder-Stage-1-UpConv	2-D Transposed Convolution	56	56	256	1	SSCB	524,544
24	Decoder-Stage-1-UpReLU	ReLU	56	56	256	1	SSCB	0
25	Decoder-Stage-1-DepthConcatenation	Depth concatenation	56	56	512	1	SSCB	0
26	Decoder-Stage-1-Conv-1	2-D Convolution	56	56	256	1	SSCB	1,179,904
27	Decoder-Stage-1-ReLU-1	ReLU	56	56	256	1	SSCB	0
28	Decoder-Stage-1-Conv-2	2-D Convolution	56	56	256	1	SSCB	590,080
29	Decoder-Stage-1-ReLU-2	ReLU	56	56	256	1	SSCB	0
30	Decoder-Stage-2-UpConv	2-D Transposed Convolution	112	112	128	1	SSCB	131,200
31	Decoder-Stage-2-UpReLU	ReLU	112	112	128	1	SSCB	0
32	Decoder-Stage-2-DepthConcatenation	Depth concatenation	112	112	256	1	SSCB	0
33	Decoder-Stage-2-Conv-1	2-D Convolution	112	112	128	1	SSCB	295,040
34	Decoder-Stage-2-ReLU-1	ReLU	112	112	128	1	SSCB	0
35	Decoder-Stage-2-Conv-2	2-D Convolution	112	112	128	1	SSCB	147,584
36	Decoder-Stage-2-ReLU-2	ReLU	112	112	128	1	SSCB	0
37	Decoder-Stage-3-UpConv	2-D Transposed Convolution	224	224	64	1	SSCB	32,832
38	Decoder-Stage-3-UpReLU	ReLU	224	224	64	1	SSCB	0
39	Decoder-Stage-3-DepthConcatenation	Depth concatenation	224	224	128	1	SSCB	0
40	Decoder-Stage-3-Conv-1	2-D Convolution	224	224	64	1	SSCB	73,792
41	Decoder-Stage-3-ReLU-1	ReLU	224	224	64	1	SSCB	0
42	Decoder-Stage-3-Conv-2	2-D Convolution	224	224	64	1	SSCB	36,928
43	Decoder-Stage-3-ReLU-2	ReLU	224	224	64	1	SSCB	0
44	Final-ConvolutionLayer	2-D Convolution	224	224	2	1	SSCB	130
45	Softmax-Layer	Softmax	224	224	2	1	SSCB	0
46	Segmentation-Layer	Pixel Classification Layer	224	224	2	1	SSCB	0

Table A2. SegNet Layer information (Activation format: S—Spatial; C–Channel; B—Batch).

	Name	Type	Activation					Learnables
1	conv1_1	2-D Convolution	224	224	64	1	SSCB	1792
2	bn_conv1_1	Batch Normalization	224	224	64	1	SSCB	128
3	relu1_1	ReLU	224	224	64	1	SSCB	0
4	conv1_2	2-D Convolution	224	224	64	1	SSCB	36,928
5	bn_conv1_2	Batch Normalization	224	224	64	1	SSCB	128
6	relu1_2	ReLU	224	224	64	1	SSCB	0
7	pool1	2-D Max Pooling						0
8	conv2_1	2-D Convolution	112	112	128	1	SSCB	73,856
9	bn_conv2_1	Batch Normalization	112	112	128	1	SSCB	256

10	relu2_1	ReLU	112	112	128	1	SSCB	0
11	conv2_2	2-D Convolution	112	112	128	1	SSCB	147,584
12	bn_conv2_2	Batch Normalization	112	112	128	1	SSCB	256
13	relu2_2	ReLU	112	112	128	1	SSCB	0
14	pool2	2-D Max Pooling						0
15	conv3_1	2-D Convolution	56	56	256	1	SSCB	295,168
16	bn_conv3_1	Batch Normalization	56	56	256	1	SSCB	512
17	relu3_1	ReLU	56	56	256	1	SSCB	0
18	conv3_2	2-D Convolution	56	56	256	1	SSCB	590,080
19	bn_conv3_2	Batch Normalization	56	56	256	1	SSCB	512
20	relu3_2	ReLU	56	56	256	1	SSCB	0
21	conv3_3	2-D Convolution	56	56	256	1	SSCB	590,080
22	bn_conv3_3	Batch Normalization	56	56	256	1	SSCB	512
23	relu3_3	ReLU	56	56	256	1	SSCB	0
24	pool3	2-D Max Pooling						0
25	conv4_1	2-D Convolution	28	28	512	1	SSCB	1,180,160
26	bn_conv4_1	Batch Normalization	28	28	512	1	SSCB	1024
27	relu4_1	ReLU	28	28	512	1	SSCB	0
28	conv4_2	2-D Convolution	28	28	512	1	SSCB	2,359,808
29	bn_conv4_2	Batch Normalization	28	28	512	1	SSCB	1024
30	relu4_2	ReLU	28	28	512	1	SSCB	0
31	conv4_3	2-D Convolution	28	28	512	1	SSCB	2,359,808
32	bn_conv4_3	Batch Normalization	28	28	512	1	SSCB	1024
33	relu4_3	ReLU	28	28	512	1	SSCB	0
34	pool4	2-D Max Pooling						0
35	conv5_1	2-D Convolution	14	14	512	1	SSCB	2,359,808
36	bn_conv5_1	Batch Normalization	14	14	512	1	SSCB	1024
37	relu5_1	ReLU	14	14	512	1	SSCB	0
38	conv5_2	2-D Convolution	14	14	512	1	SSCB	2,359,808
39	bn_conv5_2	Batch Normalization	14	14	512	1	SSCB	1024
40	relu5_2	ReLU	14	14	512	1	SSCB	0
41	conv5_3	2-D Convolution	14	14	512	1	SSCB	2,359,808
42	bn_conv5_3	Batch Normalization	14	14	512	1	SSCB	1024
43	relu5_3	ReLU	14	14	512	1	SSCB	0
44	pool5	2-D Max Pooling						0
45	decoder5_unpool	2-D Max Unpooling	14	14	512	1	SSCB	0
46	decoder5_conv3	2-D Convolution	14	14	512	1	SSCB	2,359,808
47	decoder5_bn_3	Batch Normalization	14	14	512	1	SSCB	1024
48	decoder5_relu_3	ReLU	14	14	512	1	SSCB	0
49	decoder5_conv2	2-D Convolution	14	14	512	1	SSCB	2,359,808
50	decoder5_bn_2	Batch Normalization	14	14	512	1	SSCB	1024
51	decoder5_relu_2	ReLU	14	14	512	1	SSCB	0
52	decoder5_conv1	2-D Convolution	14	14	512	1	SSCB	2,359,808
53	decoder5_bn_1	Batch Normalization	14	14	512	1	SSCB	1024

54	decoder5_relu_1	ReLU	14	14	512	1	SSCB	0
55	decoder4_unpool	2-D Max Unpooling	28	28	512	1	SSCB	0
56	decoder4_conv3	2-D Convolution	28	28	512	1	SSCB	2,359,808
57	decoder4_bn_3	Batch Normalization	28	28	512	1	SSCB	1024
58	decoder4_relu_3	ReLU	28	28	512	1	SSCB	0
59	decoder4_conv2	2-D Convolution	28	28	512	1	SSCB	2,359,808
60	decoder4_bn_2	Batch Normalization	28	28	512	1	SSCB	1024
61	decoder4_relu_2	ReLU	28	28	512	1	SSCB	0
62	decoder4_conv1	2-D Convolution	28	28	256	1	SSCB	1,179,904
63	decoder4_bn_1	Batch Normalization	28	28	256	1	SSCB	512
64	decoder4_relu_1	ReLU	28	28	256	1	SSCB	0
65	decoder3_unpool	2-D Max Unpooling	56	56	256	1	SSCB	0
66	decoder3_conv3	2-D Convolution	56	56	256	1	SSCB	590,080
67	decoder3_bn_3	Batch Normalization	56	56	256	1	SSCB	512
68	decoder3_relu_3	ReLU	56	56	256	1	SSCB	0
69	decoder3_conv2	2-D Convolution	56	56	256	1	SSCB	590,080
70	decoder3_bn_2	Batch Normalization	56	56	256	1	SSCB	512
71	decoder3_relu_2	ReLU	56	56	256	1	SSCB	0
72	decoder3_conv1	2-D Convolution	56	56	128	1	SSCB	295,040
73	decoder3_bn_1	Batch Normalization	56	56	128	1	SSCB	256
74	decoder3_relu_1	ReLU	56	56	128	1	SSCB	0
75	decoder2_unpool	2-D Max Unpooling	112	112	128	1	SSCB	0
76	decoder2_conv2	2-D Convolution	112	112	128	1	SSCB	147,584
77	decoder2_bn_2	Batch Normalization	112	112	128	1	SSCB	256
78	decoder2_relu_2	ReLU	112	112	128	1	SSCB	0
79	decoder2_conv1	2-D Convolution	112	112	64	1	SSCB	73,792
80	decoder2_bn_1	Batch Normalization	112	112	64	1	SSCB	128
81	decoder2_relu_1	ReLU	112	112	64	1	SSCB	0
82	decoder1_unpool	2-D Max Unpooling	224	224	64	1	SSCB	0
83	decoder1_conv2	2-D Convolution	224	224	64	1	SSCB	36,928
84	decoder1_bn_2	Batch Normalization	224	224	64	1	SSCB	128
85	decoder1_relu_2	ReLU	224	224	64	1	SSCB	0
86	decoder1_conv1	2-D Convolution	224	224	2	1	SSCB	1154
87	decoder1_bn_1	Batch Normalization	224	224	2	1	SSCB	4
88	decoder1_relu_1	ReLU	224	224	2	1	SSCB	0
89	softmax	Softmax	224	224	2	1	SSCB	0
90	pixelLabels	Pixel Classification Layer	224	224	2	1	SSCB	0

Table A3. DeepLabv3+/ResNet50 Layer information (Activation format: S—Spatial; C–Channel; B—Batch).

	Name	**Type**	**Activations**					**Learnables**
1	input_1	Image Input	224	224	3	1	SSCB	0
2	conv1	2-D Convolution	112	112	64	1	SSCB	9472
3	bn_conv1	Batch Normalization	112	112	64	1	SSCB	128
4	activation_1_relu	ReLU	112	112	64	1	SSCB	0
5	max_pooling2d_1	2-D Max Pooling	56	56	64	1	SSCB	0
6	res2a_branch2a	2-D Convolution	56	56	64	1	SSCB	4160
7	bn2a_branch2a	Batch Normalization	56	56	64	1	SSCB	128
8	activation_2_relu	ReLU	56	56	64	1	SSCB	0
9	res2a_branch2b	2-D Convolution	56	56	64	1	SSCB	36,928
10	bn2a_branch2b	Batch Normalization	56	56	64	1	SSCB	128
11	activation_3_relu	ReLU	56	56	64	1	SSCB	0
12	res2a_branch2c	2-D Convolution	56	56	256	1	SSCB	16,640
13	res2a_branch1	2-D Convolution	56	56	256	1	SSCB	16,640
14	bn2a_branch2c	Batch Normalization	56	56	256	1	SSCB	512
15	bn2a_branch1	Batch Normalization	56	56	256	1	SSCB	512
16	add_1	Addition	56	56	256	1	SSCB	0
17	activation_4_relu	ReLU	56	56	256	1	SSCB	0
18	res2b_branch2a	2-D Convolution	56	56	64	1	SSCB	16,448
19	bn2b_branch2a	Batch Normalization	56	56	64	1	SSCB	128
20	activation_5_relu	ReLU	56	56	64	1	SSCB	0
21	res2b_branch2b	2-D Convolution	56	56	64	1	SSCB	36,928
22	bn2b_branch2b	Batch Normalization	56	56	64	1	SSCB	128
23	activation_6_relu	ReLU	56	56	64	1	SSCB	0
24	res2b_branch2c	2-D Convolution	56	56	256	1	SSCB	16,640
25	bn2b_branch2c	Batch Normalization	56	56	256	1	SSCB	512
26	add_2	Addition	56	56	256	1	SSCB	0
27	activation_7_relu	ReLU	56	56	256	1	SSCB	0
28	res2c_branch2a	2-D Convolution	56	56	64	1	SSCB	16,448
29	bn2c_branch2a	Batch Normalization	56	56	64	1	SSCB	128
30	activation_8_relu	ReLU	56	56	64	1	SSCB	0
31	res2c_branch2b	2-D Convolution	56	56	64	1	SSCB	36,928
32	bn2c_branch2b	Batch Normalization	56	56	64	1	SSCB	128
33	activation_9_relu	ReLU	56	56	64	1	SSCB	0
34	res2c_branch2c	2-D Convolution	56	56	256	1	SSCB	16,640
35	bn2c_branch2c	Batch Normalization	56	56	256	1	SSCB	512
36	add_3	Addition	56	56	256	1	SSCB	0
37	activation_10_relu	ReLU	56	56	256	1	SSCB	0
38	res3a_branch2a	2-D Convolution	28	28	128	1	SSCB	32,896
39	bn3a_branch2a	Batch Normalization	28	28	128	1	SSCB	256
40	activation_11_relu	ReLU	28	28	128	1	SSCB	0
41	res3a_branch2b	2-D Convolution	28	28	128	1	SSCB	147,584

42	bn3a_branch2b	Batch Normalization	28	28	128	1	SSCB	256
43	activation_12_relu	ReLU	28	28	128	1	SSCB	0
44	res3a_branch2c	2-D Convolution	28	28	512	1	SSCB	66,048
45	res3a_branch1	2-D Convolution	28	28	512	1	SSCB	131,584
46	bn3a_branch2c	Batch Normalization	28	28	512	1	SSCB	1024
47	bn3a_branch1	Batch Normalization	28	28	512	1	SSCB	1024
48	add_4	Addition	28	28	512	1	SSCB	0
49	activation_13_relu	ReLU	28	28	512	1	SSCB	0
50	res3b_branch2a	2-D Convolution	28	28	128	1	SSCB	65,664
51	bn3b_branch2a	Batch Normalization	28	28	128	1	SSCB	256
52	activation_14_relu	ReLU	28	28	128	1	SSCB	0
53	res3b_branch2b	2-D Convolution	28	28	128	1	SSCB	147,584
54	bn3b_branch2b	Batch Normalization	28	28	128	1	SSCB	256
55	activation_15_relu	ReLU	28	28	128	1	SSCB	0
56	res3b_branch2c	2-D Convolution	28	28	512	1	SSCB	66,048
57	bn3b_branch2c	Batch Normalization	28	28	512	1	SSCB	1024
58	add_5	Addition	28	28	512	1	SSCB	0
59	activation_16_relu	ReLU	28	28	512	1	SSCB	0
60	res3c_branch2a	2-D Convolution	28	28	128	1	SSCB	65,664
61	bn3c_branch2a	Batch Normalization	28	28	128	1	SSCB	256
62	activation_17_relu	ReLU	28	28	128	1	SSCB	0
63	res3c_branch2b	2-D Convolution	28	28	128	1	SSCB	147,584
64	bn3c_branch2b	Batch Normalization	28	28	128	1	SSCB	256
65	activation_18_relu	ReLU	28	28	128	1	SSCB	0
66	res3c_branch2c	2-D Convolution	28	28	512	1	SSCB	66,048
67	bn3c_branch2c	Batch Normalization	28	28	512	1	SSCB	1024
68	add_6	Addition	28	28	512	1	SSCB	0
69	activation_19_relu	ReLU	28	28	512	1	SSCB	0
70	res3d_branch2a	2-D Convolution	28	28	128	1	SSCB	65,664
71	bn3d_branch2a	Batch Normalization	28	28	128	1	SSCB	256
72	activation_20_relu	ReLU	28	28	128	1	SSCB	0
73	res3d_branch2b	2-D Convolution	28	28	128	1	SSCB	147,584
74	bn3d_branch2b	Batch Normalization	28	28	128	1	SSCB	256
75	activation_21_relu	ReLU	28	28	128	1	SSCB	0
76	res3d_branch2c	2-D Convolution	28	28	512	1	SSCB	66,048
77	bn3d_branch2c	Batch Normalization	28	28	512	1	SSCB	1024
78	add_7	Addition	28	28	512	1	SSCB	0
79	activation_22_relu	ReLU	28	28	512	1	SSCB	0
80	res4a_branch2a	2-D Convolution	14	14	256	1	SSCB	131,328
81	bn4a_branch2a	Batch Normalization	14	14	256	1	SSCB	512
82	activation_23_relu	ReLU	14	14	256	1	SSCB	0
83	res4a_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
84	bn4a_branch2b	Batch Normalization	14	14	256	1	SSCB	512
85	activation_24_relu	ReLU	14	14	256	1	SSCB	0

86	res4a_branch2c	2-D Convolution	14	14	1024	1	SSCB	263,168
87	res4a_branch1	2-D Convolution	14	14	1024	1	SSCB	525,312
88	bn4a_branch2c	Batch Normalization	14	14	1024	1	SSCB	2048
89	bn4a_branch1	Batch Normalization	14	14	1024	1	SSCB	2048
90	add_8	Addition	14	14	1024	1	SSCB	0
91	activation_25_relu	ReLU	14	14	1024	1	SSCB	0
92	res4b_branch2a	2-D Convolution	14	14	256	1	SSCB	262,400
93	bn4b_branch2a	Batch Normalization	14	14	256	1	SSCB	512
94	activation_26_relu	ReLU	14	14	256	1	SSCB	0
95	res4b_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
96	bn4b_branch2b	Batch Normalization	14	14	256	1	SSCB	512
97	activation_27_relu	ReLU	14	14	256	1	SSCB	0
98	res4b_branch2c	2-D Convolution	14	14	1024	1	SSCB	263,168
99	bn4b_branch2c	Batch Normalization	14	14	1024	1	SSCB	2048
100	add_9	Addition	14	14	1024	1	SSCB	0
101	activation_28_relu	ReLU	14	14	1024	1	SSCB	0
102	res4c_branch2a	2-D Convolution	14	14	256	1	SSCB	262,400
103	bn4c_branch2a	Batch Normalization	14	14	256	1	SSCB	512
104	activation_29_relu	ReLU	14	14	256	1	SSCB	0
105	res4c_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
106	bn4c_branch2b	Batch Normalization	14	14	256	1	SSCB	512
107	activation_30_relu	ReLU	14	14	256	1	SSCB	0
108	res4c_branch2c	2-D Convolution	14	14	1024	1	SSCB	263,168
109	bn4c_branch2c	Batch Normalization	14	14	1024	1	SSCB	2048
110	add_10	Addition	14	14	1024	1	SSCB	0
111	activation_31_relu	ReLU	14	14	1024	1	SSCB	0
112	res4d_branch2a	2-D Convolution	14	14	256	1	SSCB	262,400
113	bn4d_branch2a	Batch Normalization	14	14	256	1	SSCB	512
114	activation_32_relu	ReLU	14	14	256	1	SSCB	0
115	res4d_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
116	bn4d_branch2b	Batch Normalization	14	14	256	1	SSCB	512
117	activation_33_relu	ReLU	14	14	256	1	SSCB	0
118	res4d_branch2c	2-D Convolution	14	14	1024	1	SSCB	263,168
119	bn4d_branch2c	Batch Normalization	14	14	1024	1	SSCB	2048
120	add_11	Addition	14	14	1024	1	SSCB	0
121	activation_34_relu	ReLU	14	14	1024	1	SSCB	0
122	res4e_branch2a	2-D Convolution	14	14	256	1	SSCB	262,400
123	bn4e_branch2a	Batch Normalization	14	14	256	1	SSCB	512
124	activation_35_relu	ReLU	14	14	256	1	SSCB	0
125	res4e_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
126	bn4e_branch2b	Batch Normalization	14	14	256	1	SSCB	512
127	activation_36_relu	ReLU	14	14	256	1	SSCB	0
128	res4e_branch2c	2-D Convolution	14	14	1024	1	SSCB	263,168
129	bn4e_branch2c	Batch Normalization	14	14	1024	1	SSCB	2048

130	add_12	Addition	14	14	1024	1	SSCB	0
131	activation_37_relu	ReLU	14	14	1024	1	SSCB	0
132	res4f_branch2a	2-D Convolution	14	14	256	1	SSCB	262,400
133	bn4f_branch2a	Batch Normalization	14	14	256	1	SSCB	512
134	activation_38_relu	ReLU	14	14	256	1	SSCB	0
135	res4f_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
136	bn4f_branch2b	Batch Normalization	14	14	256	1	SSCB	512
137	activation_39_relu	ReLU	14	14	256	1	SSCB	0
138	res4f_branch2c	2-D Convolution	14	14	1024	1	SSCB	263,168
139	bn4f_branch2c	Batch Normalization	14	14	1024	1	SSCB	2048
140	add_13	Addition	14	14	1024	1	SSCB	0
141	activation_40_relu	ReLU	14	14	1024	1	SSCB	0
142	res5a_branch2a	2-D Convolution	14	14	512	1	SSCB	524,800
143	bn5a_branch2a	Batch Normalization	14	14	512	1	SSCB	1024
144	activation_41_relu	ReLU	14	14	512	1	SSCB	0
145	res5a_branch2b	2-D Convolution	14	14	512	1	SSCB	2,359,808
146	bn5a_branch2b	Batch Normalization	14	14	512	1	SSCB	1024
147	activation_42_relu	ReLU	14	14	512	1	SSCB	0
148	res5a_branch2c	2-D Convolution	14	14	2048	1	SSCB	1,050,624
149	res5a_branch1	2-D Convolution	14	14	2048	1	SSCB	2,099,200
150	bn5a_branch2c	Batch Normalization	14	14	2048	1	SSCB	4096
151	bn5a_branch1	Batch Normalization	14	14	2048	1	SSCB	4096
152	add_14	Addition	14	14	2048	1	SSCB	0
153	activation_43_relu	ReLU	14	14	2048	1	SSCB	0
154	res5b_branch2a	2-D Convolution	14	14	512	1	SSCB	1,049,088
155	bn5b_branch2a	Batch Normalization	14	14	512	1	SSCB	1024
156	activation_44_relu	ReLU	14	14	512	1	SSCB	0
157	res5b_branch2b	2-D Convolution	14	14	512	1	SSCB	2,359,808
158	bn5b_branch2b	Batch Normalization	14	14	512	1	SSCB	1024
159	activation_45_relu	ReLU	14	14	512	1	SSCB	0
160	res5b_branch2c	2-D Convolution	14	14	2048	1	SSCB	1,050,624
161	bn5b_branch2c	Batch Normalization	14	14	2048	1	SSCB	4096
162	add_15	Addition	14	14	2048	1	SSCB	0
163	activation_46_relu	ReLU	14	14	2048	1	SSCB	0
164	res5c_branch2a	2-D Convolution	14	14	512	1	SSCB	1,049,088
165	bn5c_branch2a	Batch Normalization	14	14	512	1	SSCB	1024
166	activation_47_relu	ReLU	14	14	512	1	SSCB	0
167	res5c_branch2b	2-D Convolution	14	14	512	1	SSCB	2,359,808
168	bn5c_branch2b	Batch Normalization	14	14	512	1	SSCB	1024
169	activation_48_relu	ReLU	14	14	512	1	SSCB	0
170	res5c_branch2c	2-D Convolution	14	14	2048	1	SSCB	1,050,624
171	bn5c_branch2c	Batch Normalization	14	14	2048	1	SSCB	4096
172	add_16	Addition	14	14	2048	1	SSCB	0
173	activation_49_relu	ReLU	14	14	2048	1	SSCB	0

174	aspp_Conv_1	2-D Convolution	14	14	256	1	SSCB	524,544
175	aspp_BatchNorm_1	Batch Normalization	14	14	256	1	SSCB	512
176	aspp_Relu_1	ReLU	14	14	256	1	SSCB	0
177	aspp_Conv_2	2-D Convolution	14	14	256	1	SSCB	4,718,848
178	aspp_BatchNorm_2	Batch Normalization	14	14	256	1	SSCB	512
179	aspp_Relu_2	ReLU	14	14	256	1	SSCB	0
180	aspp_Conv_3	2-D Convolution	14	14	256	1	SSCB	4,718,848
181	aspp_BatchNorm_3	Batch Normalization	14	14	256	1	SSCB	512
182	aspp_Relu_3	ReLU	14	14	256	1	SSCB	0
183	aspp_Conv_4	2-D Convolution	14	14	256	1	SSCB	4,718,848
184	aspp_BatchNorm_4	Batch Normalization	14	14	256	1	SSCB	512
185	aspp_Relu_4	ReLU	14	14	256	1	SSCB	0
186	catAspp	Depth concatenation	14	14	1024	1	SSCB	0
187	dec_c1	2-D Convolution	14	14	256	1	SSCB	262,400
188	dec_bn1	Batch Normalization	14	14	256	1	SSCB	512
189	dec_relu1	ReLU	14	14	256	1	SSCB	0
190	dec_upsample1	2-D Transposed Convolution	56	56	256	1	SSCB	4,194,560
191	dec_c2	2-D Convolution	56	56	48	1	SSCB	12,336
192	dec_bn2	Batch Normalization	56	56	48	1	SSCB	96
193	dec_relu2	ReLU	56	56	48	1	SSCB	0
194	dec_crop1	Crop 2D	56	56	256	1	SSCB	0
195	dec_cat1	Depth concatenation	56	56	304	1	SSCB	0
196	dec_c3	2-D Convolution	56	56	256	1	SSCB	700,672
197	dec_bn3	Batch Normalization	56	56	256	1	SSCB	512
198	dec_relu3	ReLU	56	56	256	1	SSCB	0
199	dec_c4	2-D Convolution	56	56	256	1	SSCB	590,080
200	dec_bn4	Batch Normalization	56	56	256	1	SSCB	512
201	dec_relu4	ReLU	56	56	256	1	SSCB	0
202	scorer	2-D Convolution	56	56	2	1	SSCB	514
203	dec_upsample2	2-D Transposed Convolution	224	224	2	1	SSCB	258
204	dec_crop2	Crop 2D	224	224	2	1	SSCB	0
205	softmax-out	Softmax	224	224	2	1	SSCB	0
206	labels	Pixel Classification Layer	224	224	2	1	SSCB	0

Table A4. DeepLabv3+/ResNet18 Layer information (Activation format: S—Spatial; C–Channel; B—Batch).

	Name	Type	Activations					Learnables
1	data	Image Input	224	224	3	1	SSCB	0
2	conv1	2-D Convolution	112	112	64	1	SSCB	9472
3	bn_conv1	Batch Normalization	112	112	64	1	SSCB	128
4	conv1_relu	ReLU	112	112	64	1	SSCB	0
5	pool1	2-D Max Pooling	56	56	64	1	SSCB	0
6	res2a_branch2a	2-D Convolution	56	56	64	1	SSCB	36,928
7	bn2a_branch2a	Batch Normalization	56	56	64	1	SSCB	128

8	res2a_branch2a_relu	ReLU	56	56	64	1	SSCB	0
9	res2a_branch2b	2-D Convolution	56	56	64	1	SSCB	36,928
10	bn2a_branch2b	Batch Normalization	56	56	64	1	SSCB	128
11	res2a	Addition	56	56	64	1	SSCB	0
12	res2a_relu	ReLU	56	56	64	1	SSCB	0
13	res2b_branch2a	2-D Convolution	56	56	64	1	SSCB	36,928
14	bn2b_branch2a	Batch Normalization	56	56	64	1	SSCB	128
15	res2b_branch2a_relu	ReLU	56	56	64	1	SSCB	0
16	res2b_branch2b	2-D Convolution	56	56	64	1	SSCB	36,928
17	bn2b_branch2b	Batch Normalization	56	56	64	1	SSCB	128
18	res2b	Addition	56	56	64	1	SSCB	0
19	res2b_relu	ReLU	56	56	64	1	SSCB	0
20	res3a_branch2a	2-D Convolution	28	28	128	1	SSCB	73,856
21	bn3a_branch2a	Batch Normalization	28	28	128	1	SSCB	256
22	res3a_branch2a_relu	ReLU	28	28	128	1	SSCB	0
23	res3a_branch2b	2-D Convolution	28	28	128	1	SSCB	147,584
24	bn3a_branch2b	Batch Normalization	28	28	128	1	SSCB	256
25	res3a_branch1	2-D Convolution	28	28	128	1	SSCB	8320
26	bn3a_branch1	Batch Normalization	28	28	128	1	SSCB	256
27	res3a	Addition	28	28	128	1	SSCB	0
28	res3a_relu	ReLU	28	28	128	1	SSCB	0
29	res3b_branch2a	2-D Convolution	28	28	128	1	SSCB	147,584
30	bn3b_branch2a	Batch Normalization	28	28	128	1	SSCB	256
31	res3b_branch2a_relu	ReLU	28	28	128	1	SSCB	0
32	res3b_branch2b	2-D Convolution	28	28	128	1	SSCB	147,584
33	bn3b_branch2b	Batch Normalization	28	28	128	1	SSCB	256
34	res3b	Addition	28	28	128	1	SSCB	0
35	res3b_relu	ReLU	28	28	128	1	SSCB	0
36	res4a_branch2a	2-D Convolution	14	14	256	1	SSCB	295,168
37	bn4a_branch2a	Batch Normalization	14	14	256	1	SSCB	512
38	res4a_branch2a_relu	ReLU	14	14	256	1	SSCB	0
39	res4a_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
40	bn4a_branch2b	Batch Normalization	14	14	256	1	SSCB	512
41	res4a_branch1	2-D Convolution	14	14	256	1	SSCB	33,024
42	bn4a_branch1	Batch Normalization	14	14	256	1	SSCB	512
43	res4a	Addition	14	14	256	1	SSCB	0
44	res4a_relu	ReLU	14	14	256	1	SSCB	0
45	res4b_branch2a	2-D Convolution	14	14	256	1	SSCB	590,080
46	bn4b_branch2a	Batch Normalization	14	14	256	1	SSCB	512
47	res4b_branch2a_relu	ReLU	14	14	256	1	SSCB	0
48	res4b_branch2b	2-D Convolution	14	14	256	1	SSCB	590,080
49	bn4b_branch2b	Batch Normalization	14	14	256	1	SSCB	512
50	res4b	Addition	14	14	256	1	SSCB	0
51	res4b_relu	ReLU	14	14	256	1	SSCB	0

52	res5a_branch2a	2-D Convolution	14	14	512	1	SSCB	1,180,160
53	bn5a_branch2a	Batch Normalization	14	14	512	1	SSCB	1024
54	res5a_branch2a_relu	ReLU	14	14	512	1	SSCB	0
55	res5a_branch2b	2-D Convolution	14	14	512	1	SSCB	2,359,808
56	bn5a_branch2b	Batch Normalization	14	14	512	1	SSCB	1024
57	res5a_branch1	2-D Convolution	14	14	512	1	SSCB	131,584
58	bn5a_branch1	Batch Normalization	14	14	512	1	SSCB	1024
59	res5a	Addition	14	14	512	1	SSCB	0
60	res5a_relu	ReLU	14	14	512	1	SSCB	0
61	res5b_branch2a	2-D Convolution	14	14	512	1	SSCB	2,359,808
62	bn5b_branch2a	Batch Normalization	14	14	512	1	SSCB	1024
63	res5b_branch2a_relu	ReLU	14	14	512	1	SSCB	0
64	res5b_branch2b	2-D Convolution	14	14	512	1	SSCB	2,359,808
65	bn5b_branch2b	Batch Normalization	14	14	512	1	SSCB	1024
66	res5b	Addition	14	14	512	1	SSCB	0
67	res5b_relu	ReLU	14	14	512	1	SSCB	0
68	aspp_Conv_1	2-D Convolution	14	14	256	1	SSCB	131,328
69	aspp_BatchNorm_1	Batch Normalization	14	14	256	1	SSCB	512
70	aspp_Relu_1	ReLU	14	14	256	1	SSCB	0
71	aspp_Conv_2	2-D Convolution	14	14	256	1	SSCB	1,179,904
72	aspp_BatchNorm_2	Batch Normalization	14	14	256	1	SSCB	512
73	aspp_Relu_2	ReLU	14	14	256	1	SSCB	0
74	aspp_Conv_3	2-D Convolution	14	14	256	1	SSCB	1,179,904
75	aspp_BatchNorm_3	Batch Normalization	14	14	256	1	SSCB	512
76	aspp_Relu_3	ReLU	14	14	256	1	SSCB	0
77	aspp_Conv_4	2-D Convolution	14	14	256	1	SSCB	1,179,904
78	aspp_BatchNorm_4	Batch Normalization	14	14	256	1	SSCB	512
79	aspp_Relu_4	ReLU	14	14	256	1	SSCB	0
80	catAspp	Depth concatenation	14	14	###	1	SSCB	0
81	dec_c1	2-D Convolution	14	14	256	1	SSCB	262,400
82	dec_bn1	Batch Normalization	14	14	256	1	SSCB	512
83	dec_relu1	ReLU	14	14	256	1	SSCB	0
84	dec_upsample1	2-D Transposed Convolution	56	56	256	1	SSCB	4,194,560
85	dec_c2	2-D Convolution	56	56	48	1	SSCB	3120
86	dec_bn2	Batch Normalization	56	56	48	1	SSCB	96
87	dec_relu2	ReLU	56	56	48	1	SSCB	0
88	dec_crop1	Crop 2D	56	56	256	1	SSCB	0
89	dec_cat1	Depth concatenation	56	56	304	1	SSCB	0
90	dec_c3	2-D Convolution	56	56	256	1	SSCB	700,672
91	dec_bn3	Batch Normalization	56	56	256	1	SSCB	512
92	dec_relu3	ReLU	56	56	256	1	SSCB	0
93	dec_c4	2-D Convolution	56	56	256	1	SSCB	590,080
94	dec_bn4	Batch Normalization	56	56	256	1	SSCB	512
95	dec_relu4	ReLU	56	56	256	1	SSCB	0

96	scorer	2-D Convolution	56	56	2	1	SSCB	514
97	dec_upsample2	2-D Transposed Convolution	224	224	2	1	SSCB	258
98	dec_crop2	Crop 2D	224	224	2	1	SSCB	0
99	softmax-out	Softmax	224	224	2	1	SSCB	0
100	labels	Pixel Classification Layer	224	224	2	1	SSCB	0

Table A5. DeepLabv3+/MobileNetv2 Layer information (Activation format: S—Spatial; C–Channel; B—Batch).

	Name	Type	Activations					Learnables
1	input_1	Image Input	224	224	3	1	SSCB	0
2	Conv1	2-D Convolution	112	112	32	1	SSCB	896
3	bn_Conv1	Batch Normalization	112	112	32	1	SSCB	64
4	Conv1_relu	Clipped ReLU	112	112	32	1	SSCB	0
5	expanded_conv_depthwise	2-D Grouped Convolution	112	112	32	1	SSCB	320
6	expanded_conv_depthwise_BN	Batch Normalization	112	112	32	1	SSCB	64
7	expanded_conv_depthwise_relu	Clipped ReLU	112	112	32	1	SSCB	0
8	expanded_conv_project	2-D Convolution	112	112	16	1	SSCB	528
9	expanded_conv_project_BN	Batch Normalization	112	112	16	1	SSCB	32
10	block_1_expand	2-D Convolution	112	112	96	1	SSCB	1632
11	block_1_expand_BN	Batch Normalization	112	112	96	1	SSCB	192
12	block_1_expand_relu	Clipped ReLU	112	112	96	1	SSCB	0
13	block_1_depthwise	2-D Grouped Convolution	56	56	96	1	SSCB	960
14	block_1_depthwise_BN	Batch Normalization	56	56	96	1	SSCB	192
15	block_1_depthwise_relu	Clipped ReLU	56	56	96	1	SSCB	0
16	block_1_project	2-D Convolution	56	56	24	1	SSCB	2328
17	block_1_project_BN	Batch Normalization	56	56	24	1	SSCB	48
18	block_2_expand	2-D Convolution	56	56	144	1	SSCB	3600
19	block_2_expand_BN	Batch Normalization	56	56	144	1	SSCB	288
20	block_2_expand_relu	Clipped ReLU	56	56	144	1	SSCB	0
21	block_2_depthwise	2-D Grouped Convolution	56	56	144	1	SSCB	1440
22	block_2_depthwise_BN	Batch Normalization	56	56	144	1	SSCB	288
23	block_2_depthwise_relu	Clipped ReLU	56	56	144	1	SSCB	0
24	block_2_project	2-D Convolution	56	56	24	1	SSCB	3480
25	block_2_project_BN	Batch Normalization	56	56	24	1	SSCB	48
26	block_2_add	Addition	56	56	24	1	SSCB	0
27	block_3_expand	2-D Convolution	56	56	144	1	SSCB	3600
28	block_3_expand_BN	Batch Normalization	56	56	144	1	SSCB	288
29	block_3_expand_relu	Clipped ReLU	56	56	144	1	SSCB	0
30	block_3_depthwise	2-D Grouped Convolution	28	28	144	1	SSCB	1440
31	block_3_depthwise_BN	Batch Normalization	28	28	144	1	SSCB	288
32	block_3_depthwise_relu	Clipped ReLU	28	28	144	1	SSCB	0
33	block_3_project	2-D Convolution	28	28	32	1	SSCB	4640
34	block_3_project_BN	Batch Normalization	28	28	32	1	SSCB	64
35	block_4_expand	2-D Convolution	28	28	192	1	SSCB	6336

36	block_4_expand_BN	Batch Normalization	28	28	192	1	SSCB	384
37	block_4_expand_relu	Clipped ReLU	28	28	192	1	SSCB	0
38	block_4_depthwise	2-D Grouped Convolution	28	28	192	1	SSCB	1920
39	block_4_depthwise_BN	Batch Normalization	28	28	192	1	SSCB	384
40	block_4_depthwise_relu	Clipped ReLU	28	28	192	1	SSCB	0
41	block_4_project	2-D Convolution	28	28	32	1	SSCB	6176
42	block_4_project_BN	Batch Normalization	28	28	32	1	SSCB	64
43	block_4_add	Addition	28	28	32	1	SSCB	0
44	block_5_expand	2-D Convolution	28	28	192	1	SSCB	6336
45	block_5_expand_BN	Batch Normalization	28	28	192	1	SSCB	384
46	block_5_expand_relu	Clipped ReLU	28	28	192	1	SSCB	0
47	block_5_depthwise	2-D Grouped Convolution	28	28	192	1	SSCB	1920
48	block_5_depthwise_BN	Batch Normalization	28	28	192	1	SSCB	384
49	block_5_depthwise_relu	Clipped ReLU	28	28	192	1	SSCB	0
50	block_5_project	2-D Convolution	28	28	32	1	SSCB	6176
51	block_5_project_BN	Batch Normalization	28	28	32	1	SSCB	64
52	block_5_add	Addition	28	28	32	1	SSCB	0
53	block_6_expand	2-D Convolution	28	28	192	1	SSCB	6336
54	block_6_expand_BN	Batch Normalization	28	28	192	1	SSCB	384
55	block_6_expand_relu	Clipped ReLU	28	28	192	1	SSCB	0
56	block_6_depthwise	2-D Grouped Convolution	14	14	192	1	SSCB	1920
57	block_6_depthwise_BN	Batch Normalization	14	14	192	1	SSCB	384
58	block_6_depthwise_relu	Clipped ReLU	14	14	192	1	SSCB	0
59	block_6_project	2-D Convolution	14	14	64	1	SSCB	12,352
60	block_6_project_BN	Batch Normalization	14	14	64	1	SSCB	128
61	block_7_expand	2-D Convolution	14	14	384	1	SSCB	24,960
62	block_7_expand_BN	Batch Normalization	14	14	384	1	SSCB	768
63	block_7_expand_relu	Clipped ReLU	14	14	384	1	SSCB	0
64	block_7_depthwise	2-D Grouped Convolution	14	14	384	1	SSCB	3840
65	block_7_depthwise_BN	Batch Normalization	14	14	384	1	SSCB	768
66	block_7_depthwise_relu	Clipped ReLU	14	14	384	1	SSCB	0
67	block_7_project	2-D Convolution	14	14	64	1	SSCB	24,640
68	block_7_project_BN	Batch Normalization	14	14	64	1	SSCB	128
69	block_7_add	Addition	14	14	64	1	SSCB	0
70	block_8_expand	2-D Convolution	14	14	384	1	SSCB	24,960
71	block_8_expand_BN	Batch Normalization	14	14	384	1	SSCB	768
72	block_8_expand_relu	Clipped ReLU	14	14	384	1	SSCB	0
73	block_8_depthwise	2-D Grouped Convolution	14	14	384	1	SSCB	3840
74	block_8_depthwise_BN	Batch Normalization	14	14	384	1	SSCB	768
75	block_8_depthwise_relu	Clipped ReLU	14	14	384	1	SSCB	0
76	block_8_project	2-D Convolution	14	14	64	1	SSCB	24,640
77	block_8_project_BN	Batch Normalization	14	14	64	1	SSCB	128
78	block_8_add	Addition	14	14	64	1	SSCB	0
79	block_9_expand	2-D Convolution	14	14	384	1	SSCB	24,960

80	block_9_expand_BN	Batch Normalization	14	14	384	1	SSCB	768
81	block_9_expand_relu	Clipped ReLU	14	14	384	1	SSCB	0
82	block_9_depthwise	2-D Grouped Convolution	14	14	384	1	SSCB	3840
83	block_9_depthwise_BN	Batch Normalization	14	14	384	1	SSCB	768
84	block_9_depthwise_relu	Clipped ReLU	14	14	384	1	SSCB	0
85	block_9_project	2-D Convolution	14	14	64	1	SSCB	24,640
86	block_9_project_BN	Batch Normalization	14	14	64	1	SSCB	128
87	block_9_add	Addition	14	14	64	1	SSCB	0
88	block_10_expand	2-D Convolution	14	14	384	1	SSCB	24,960
89	block_10_expand_BN	Batch Normalization	14	14	384	1	SSCB	768
90	block_10_expand_relu	Clipped ReLU	14	14	384	1	SSCB	0
91	block_10_depthwise	2-D Grouped Convolution	14	14	384	1	SSCB	3840
92	block_10_depthwise_BN	Batch Normalization	14	14	384	1	SSCB	768
93	block_10_depthwise_relu	Clipped ReLU	14	14	384	1	SSCB	0
94	block_10_project	2-D Convolution	14	14	96	1	SSCB	36,960
95	block_10_project_BN	Batch Normalization	14	14	96	1	SSCB	192
96	block_11_expand	2-D Convolution	14	14	576	1	SSCB	55,872
97	block_11_expand_BN	Batch Normalization	14	14	576	1	SSCB	1152
98	block_11_expand_relu	Clipped ReLU	14	14	576	1	SSCB	0
99	block_11_depthwise	2-D Grouped Convolution	14	14	576	1	SSCB	5760
100	block_11_depthwise_BN	Batch Normalization	14	14	576	1	SSCB	1152
101	block_11_depthwise_relu	Clipped ReLU	14	14	576	1	SSCB	0
102	block_11_project	2-D Convolution	14	14	96	1	SSCB	55,392
103	block_11_project_BN	Batch Normalization	14	14	96	1	SSCB	192
104	block_11_add	Addition	14	14	96	1	SSCB	0
105	block_12_expand	2-D Convolution	14	14	576	1	SSCB	55,872
106	block_12_expand_BN	Batch Normalization	14	14	576	1	SSCB	1152
107	block_12_expand_relu	Clipped ReLU	14	14	576	1	SSCB	0
108	block_12_depthwise	2-D Grouped Convolution	14	14	576	1	SSCB	5760
109	block_12_depthwise_BN	Batch Normalization	14	14	576	1	SSCB	1152
110	block_12_depthwise_relu	Clipped ReLU	14	14	576	1	SSCB	0
111	block_12_project	2-D Convolution	14	14	96	1	SSCB	55,392
112	block_12_project_BN	Batch Normalization	14	14	96	1	SSCB	192
113	block_12_add	Addition	14	14	96	1	SSCB	0
114	block_13_expand	2-D Convolution	14	14	576	1	SSCB	55,872
115	block_13_expand_BN	Batch Normalization	14	14	576	1	SSCB	1152
116	block_13_expand_relu	Clipped ReLU	14	14	576	1	SSCB	0
117	block_13_depthwise	2-D Grouped Convolution	14	14	576	1	SSCB	5760
118	block_13_depthwise_BN	Batch Normalization	14	14	576	1	SSCB	1152
119	block_13_depthwise_relu	Clipped ReLU	14	14	576	1	SSCB	0
120	block_13_project	2-D Convolution	14	14	160	1	SSCB	92,320
121	block_13_project_BN	Batch Normalization	14	14	160	1	SSCB	320
122	block_14_expand	2-D Convolution	14	14	960	1	SSCB	154,560
123	block_14_expand_BN	Batch Normalization	14	14	960	1	SSCB	1920

124	block_14_expand_relu	Clipped ReLU	14	14	960	1	SSCB	0
125	block_14_depthwise	2-D Grouped Convolution	14	14	960	1	SSCB	9600
126	block_14_depthwise_BN	Batch Normalization	14	14	960	1	SSCB	1920
127	block_14_depthwise_relu	Clipped ReLU	14	14	960	1	SSCB	0
128	block_14_project	2-D Convolution	14	14	160	1	SSCB	153,760
129	block_14_project_BN	Batch Normalization	14	14	160	1	SSCB	320
130	block_14_add	Addition	14	14	160	1	SSCB	0
131	block_15_expand	2-D Convolution	14	14	960	1	SSCB	154,560
132	block_15_expand_BN	Batch Normalization	14	14	960	1	SSCB	1920
133	block_15_expand_relu	Clipped ReLU	14	14	960	1	SSCB	0
134	block_15_depthwise	2-D Grouped Convolution	14	14	960	1	SSCB	9600
135	block_15_depthwise_BN	Batch Normalization	14	14	960	1	SSCB	1920
136	block_15_depthwise_relu	Clipped ReLU	14	14	960	1	SSCB	0
137	block_15_project	2-D Convolution	14	14	160	1	SSCB	153,760
138	block_15_project_BN	Batch Normalization	14	14	160	1	SSCB	320
139	block_15_add	Addition	14	14	160	1	SSCB	0
140	block_16_expand	2-D Convolution	14	14	960	1	SSCB	154,560
141	block_16_expand_BN	Batch Normalization	14	14	960	1	SSCB	1920
142	block_16_expand_relu	Clipped ReLU	14	14	960	1	SSCB	0
143	block_16_depthwise	2-D Grouped Convolution	14	14	960	1	SSCB	9600
144	block_16_depthwise_BN	Batch Normalization	14	14	960	1	SSCB	1920
145	block_16_depthwise_relu	Clipped ReLU	14	14	960	1	SSCB	0
146	block_16_project	2-D Convolution	14	14	320	1	SSCB	307,520
147	block_16_project_BN	Batch Normalization	14	14	320	1	SSCB	640
148	aspp_Conv_1_depthwise	2-D Grouped Convolution	14	14	320	1	SSCB	640
149	aspp_Conv_1_pointwise	2-D Convolution	14	14	256	1	SSCB	82,176
150	aspp_BatchNorm_1	Batch Normalization	14	14	256	1	SSCB	512
151	aspp_Relu_1	ReLU	14	14	256	1	SSCB	0
152	aspp_Conv_2_depthwise	2-D Grouped Convolution	14	14	320	1	SSCB	3200
153	aspp_Conv_2_pointwise	2-D Convolution	14	14	256	1	SSCB	82,176
154	aspp_BatchNorm_2	Batch Normalization	14	14	256	1	SSCB	512
155	aspp_Relu_2	ReLU	14	14	256	1	SSCB	0
156	aspp_Conv_3_depthwise	2-D Grouped Convolution	14	14	320	1	SSCB	3200
157	aspp_Conv_3_pointwise	2-D Convolution	14	14	256	1	SSCB	82,176
158	aspp_BatchNorm_3	Batch Normalization	14	14	256	1	SSCB	512
159	aspp_Relu_3	ReLU	14	14	256	1	SSCB	0
160	aspp_Conv_4_depthwise	2-D Grouped Convolution	14	14	320	1	SSCB	3200
161	aspp_Conv_4_pointwise	2-D Convolution	14	14	256	1	SSCB	82,176
162	aspp_BatchNorm_4	Batch Normalization	14	14	256	1	SSCB	512
163	aspp_Relu_4	ReLU	14	14	256	1	SSCB	0
164	catAspp	Depth concatenation	14	14	1024	1	SSCB	0
165	dec_c1	2-D Convolution	14	14	256	1	SSCB	262,400
166	dec_bn1	Batch Normalization	14	14	256	1	SSCB	512
167	dec_relu1	ReLU	14	14	256	1	SSCB	0

168	dec_upsample1	2-D Transposed Convolution	56	56	256	1	SSCB	4,194,560
169	dec_c2	2-D Convolution	56	56	48	1	SSCB	6960
170	dec_bn2	Batch Normalization	56	56	48	1	SSCB	96
171	dec_relu2	ReLU	56	56	48	1	SSCB	0
172	dec_crop1	Crop 2D	56	56	256	1	SSCB	0
173	dec_cat1	Depth concatenation	56	56	304	1	SSCB	0
174	dec_c3_depthwise	2-D Grouped Convolution	56	56	304	1	SSCB	3040
175	dec_c3_pointwise	2-D Convolution	56	56	256	1	SSCB	78,080
176	dec_bn3	Batch Normalization	56	56	256	1	SSCB	512
177	dec_relu3	ReLU	56	56	256	1	SSCB	0
178	dec_c4_depthwise	2-D Grouped Convolution	56	56	256	1	SSCB	2560
179	dec_c4_pointwise	2-D Convolution	56	56	256	1	SSCB	65,792
180	dec_bn4	Batch Normalization	56	56	256	1	SSCB	512
181	dec_relu4	ReLU	56	56	256	1	SSCB	0
182	scorer	2-D Convolution	56	56	2	1	SSCB	514
183	dec_upsample2	2-D Transposed Convolution	224	224	2	1	SSCB	258
184	dec_crop2	Crop 2D	224	224	2	1	SSCB	0
185	softmax-out	Softmax	224	224	2	1	SSCB	0
186	labels	Pixel Classification Layer	224	224	2	1	SSCB	0

References

1. Larmuseau, M.; Sluydts, M.; Theuwissen, K.; Duprez, L.; Dhaene, T.; Cottenier, S. Race against the Machine: Can Deep Learning Recognize Microstructures as Well as the Trained Human Eye? *Scr. Mater.* **2021**, *193*, 33–37. https://doi.org/10.1016/j.scriptamat.2020.10.026.
2. DeCost, B.L.; Francis, T.; Holm, E.A. Exploring the Microstructure Manifold: Image Texture Representations Applied to Ultrahigh Carbon Steel Microstructures. *Acta Mater.* **2017**, *133*, 30–40. https://doi.org/10.1016/j.actamat.2017.05.014.
3. Gupta, S.; Banerjee, A.; Sarkar, J.; Kundu, M.; Sinha, S.K.; Bandyopadhyay, N.R.; Ganguly, S. Modelling the Steel Microstructure Knowledge for In-Silico Recognition of Phases Using Machine Learning. *Mater. Chem. Phys.* **2020**, *252*, 123286. https://doi.org/10.1016/j.matchemphys.2020.123286.
4. Wang, J.; Fa, Y.; Tian, Y.; Yu, X. A Machine-Learning Approach to Predict Creep Properties of Cr–Mo Steel with Time-Temperature Parameters. *J. Mater. Res. Technol.* **2021**, *13*, 635–650. https://doi.org/10.1016/j.jmrt.2021.04.079.
5. Yucel, B.; Yucel, S.; Ray, A.; Duprez, L.; Kalidindi, S.R. Mining the Correlations between Optical Micrographs and Mechanical Properties of Cold-Rolled HSLA Steels Using Machine Learning Approaches. *Integr. Mater. Manuf. Innov.* **2020**, *9*, 240–256. https://doi.org/10.1007/s40192-020-00183-3.
6. Wang, Z.-L.; Adachi, Y. Property Prediction and Properties-to-Microstructure Inverse Analysis of Steels by a Machine-Learning Approach. *Mater. Sci. Eng. A Struct. Mater.* **2019**, *744*, 661–670. https://doi.org/10.1016/j.msea.2018.12.049.
7. Larmuseau, M.; Theuwissen, K.; Lejaeghere, K.; Duprez, L.; Dhaene, T.; Cottenier, S. Towards Accurate Processing-Structure-Property Links Using Deep Learning. *Scr. Mater.* **2022**, *211*, 114478. https://doi.org/10.1016/j.scriptamat.2021.114478.
8. Muñoz-Rodenas, J.; García-Sevilla, F.; Coello-Sobrino, J.; Martínez-Martínez, A.; Miguel-Eguía, V. Effectiveness of Machine-Learning and Deep-Learning Strategies for the Classification of Heat Treatments Applied to Low-Carbon Steels Based on Microstructural Analysis. *Appl. Sci.* **2023**, *13*, 3479. https://doi.org/10.3390/app13063479.
9. Luengo, J.; Moreno, R.; Sevillano, I.; Charte, D.; Peláez, A.; Fernández, M.; Herrera, F. A tutorial on the segmentation of metallographic images: Taxonomy, new MetalDAM dataset, deep learning-based ensemble model, experimental analysis and challenges. *Inf. Fusion* **2022**, *78*, 232–253. https://doi.org/10.1016/j.inffus.2021.09.018.
10. Bulgarevich, D.; Tsukamoto, S.; Kasuya, T.; Demura, M.; Watanabe, M. Pattern recognition with machine learning on optical microscopy images of typical metallurgical microstructures. *Sci. Rep.* **2018**, *8*, 2078. https://doi.org/10.1038/s41598-018-20438-6.
11. Bachmann, B.; Müller, M.; Britz, D.; Durmaz, A.; Ackermann, M.; Shchyglo, O.; Staudt, T.; Mücklich, F. Efficient reconstruction of prior austenite grains in steel from etched light optical micrographs using deep learning and annotations from correlative microscopy. *Front. Mater.* **2022**, *9*, 1033505. https://doi.org/10.3389/fmats.2022.1033505.
12. Han, Y.; Li, R.; Yang, S.; Chen, Q.; Wang, B.; Liu, Y. Center-environment feature models for materials image segmentation based on machine learning. *Sci. Rep.* **2022**, *12*, 12960. https://doi.org/10.1038/s41598-022-16824-w.

13. Kim, H.; Inoue, J.; Kasuya, T. Unsupervised microstructure segmentation by mimicking metallurgists' approach to pattern recognition. *Sci. Rep.* **2020**, *10*, 17835. https://doi.org/10.1038/s41598-020-74935-8.
14. Breumier, S.; Martinez, T.; Frincu, B.; Gey, N.; Couturier, A.; Loukachenko, N.; Aba-perea, P.E.; Germain, L. Leveraging EBSD data by deep learning for bainite, ferrite and martensite segmentation *Mater. Charact.* **2022**, *186*, 111805. https://doi.org/10.1016/j.matchar.2022.111805.
15. Chaurasia, N.; Jha, S.K.; Sangal, S. A Novel Training Methodology for Phase Segmentation of Steel Microstructures Using a Deep Learning Algorithm. *Materialia* **2023**, *30*, 101803. https://doi.org/10.1016/j.mtla.2023.101803.
16. Liu, J.; Cao, G.; Wang, H.; Cui, C.; Liu, Z. Development of Intelligent Methodologies Perceiving Microstructure and Mechanical Properties of Hot Rolled Steels. *Measurement* **2023**, *221*, 113526. https://doi.org/10.1016/j.measurement.2023.113526.
17. Azimi, S.M.; Britz, D.; Engstler, M.; Fritz, M.; Mücklich, F. Advanced Steel Microstructural Classification by Deep Learning Methods. *Sci. Rep.* **2018**, *8*, 2128. https://doi.org/10.1038/s41598-018-20037-5.
18. Martinez Ostormujof, T.; Purushottam Raj Purohit, R.R.P.; Breumier, S.; Gey, N.; Salib, M.; Germain, L. Deep Learning for Automated Phase Segmentation in EBSD Maps. A Case Study in Dual Phase Steel Microstructures. *Mater. Charact.* **2022**, *184*, 111638. https://doi.org/10.1016/j.matchar.2021.111638.
19. Xie, L.; Li, W.; Fan, L.; Zhou, M. Automatic Identification of the Multiphase Microstructures of Steels Based on ASPP-FCN. *Steel Res. Int.* **2023**, *94*, 202200204. https://doi.org/10.1002/srin.202200204.
20. Ma, X.; Yu, Y. Training Tricks for Steel Microstructure Segmentation with Deep Learning. *Processes* **2023**, *11*, 3298. https://doi.org/10.3390/pr11123298.
21. Bihani, A.; Daigle, H.; Santos, J.E.; Landry, C.; Prodanović, M.; Milliken, K. MudrockNet: Semantic Segmentation of Mudrock SEM Images through Deep Learning. *Comput. Geosci.* **2022**, *158*, 104952. https://doi.org/10.1016/j.cageo.2021.104952.
22. Arganda-Carreras, I.; Kaynig, V.; Rueden, C.; Eliceiri, K.W.; Schindelin, J.; Cardona, A.; Seung, H.S. Trainable Weka Segmentation: A machine learning tool for microscopy pixel classification. *Bioinformatics* **2017**, *33*, 2424–2426. https://doi.org/10.1093/bioinformatics/btx180.
23. Somasundaram, E.; Kaufman, R.; Brady, S. Advancements in Automated Tissue Segmentation Pipeline for Contrast-Enhanced CT Scans of Adult and Pediatric Patients. In Proceedings of the SPIE Medical Imaging, Orlando, FL, USA, 13–16 February 2017; Armato, S.G., Petrick, N.A., Eds.; SPIE: Bellingham, WA, USA, 2017.
24. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention (MICCAI) 2015, Munich, Germany, 5–9 October 2015; Volume 9351; pp. 234–241.
25. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*, 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615.
26. Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; Springer International Publishing: Cham, Switzerland, 2018; pp. 833–851, ISBN 9783030012335.
27. Csurka, G.; Larlus, D.; Perronnin, F. What is a good evaluation measure for semantic segmentation? In Proceedings of the British Machine Vision Conference, Bristol, UK, 9–13 September 2013; pp. 32.1–32.11.
28. Swain, B.R.; Cho, D.; Park, J.; Roh, J.-S.; Ko, J. Complex-Phase Steel Microstructure Segmentation Using UNet: Analysis across Different Magnifications and Steel Types. *Materials* **2023**, *16*, 7254. https://doi.org/10.3390/ma16237254.
29. Han, Y.; Li, R.; Wang, B.; Ruan, L.; Chen, Q. A Pseudo-Labeling Based Weakly Supervised Segmentation Method for Few-Shot Texture Images. *Expert Syst. Appl.* **2024**, *238*, 122110. https://doi.org/10.1016/j.eswa.2023.122110.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep Into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
31. Ajioka, F.; Wang, Z.-L.; Ogawa, T.; Adachi, Y. Development of High Accuracy Segmentation Model for Microstructure of Steel by Deep Learning. *ISIJ Int.* **2020**, *60*, 954–959. https://doi.org/10.2355/isijinternational.isijint-2019-568.
32. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556v6.
33. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis. IJCV* **2015**, *115*, 211–252.
34. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 770–778.
35. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 4510–4520.