Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-024-02383-6

Robust image-based cross-sectional grain boundary detection and characterization using machine learning

Nicholas Satterlee[1] · Runjian Jiang[1] · Eugene Olevsky[1] · Elisa Torresani[1] · Xiaowei Zuo[1] · John S. Kang[1]

Received: 30 August 2023 / Accepted: 19 March 2024

Abstract
Understanding the anisotropic sintering behavior of 3D-printed materials requires massive analytic studies on their grain boundary (GB) structures. Accurate characterization of the GBs is critical to study the metallurgical process. However, it is challenging and time-consuming for sintered 3D-printed materials due to immature etching and residual pores. In this study, we developed a machine learning-based method of characterizing GBs of sintered 3D-printed materials. The developed method is also generalizable and robust enough to characterize GBs from other non-3D-printed materials. This method can be applied to a small dataset because it includes a diffusion network that generate augmented images for training. The study compared various machine learning methods commonly used for segmentation, which include UNet, ResNeXt, and Ensemble of UNets. The comparison results showed that the Ensemble of UNets outperformed the other methods for the GB detection and characterization. The model is tested on unclear GBs from sintered 3D-printed samples processed with non-optimized etching and classifies the GBs with around 90% accuracy. The model is also tested on images with clear GBs from literature and classifies GBs with 92% accuracy.

Keywords Additive manufacturing · Binder jetting · Grain boundary · Porosity · Sintering · Machine learning · Materials characterization

Introduction

Grain boundaries (GBs) are an essential feature of polycrystalline materials, which significantly influence the mechanical, electrical, and thermal properties of the materials. GB detection is crucial for understanding the microstructure of materials and for developing new materials with improved properties. As close-up images of the cross-sections of the materials become available with the advancement of optical microscopy, micro-structural analysis has been applied in many different industries to identify GBs and to help understand the micro-structural behaviors of a material.

The traditional way to determine the GBs from the cross-section images was through segmenting the GBs by hand (Dengiz et al., 2005); however, such manual approach is usually prone to operator fatigue, human errors, and leads to inconsistency across analysis of images. To alleviate the burden or such repetitive tasks, traditional image processing methods such as thresholding, tolerance-based neighbor analysis, or Euclidean distance measurement were used to automatically segment the GBs from the cross-section images (Catania et al., 2022; Dengiz et al., 2005).

However, the traditional image processing methods for GB detection are time-consuming and require significant effort to obtain accurate results. With the advancement in machine learning (ML) in image processing, ML-based techniques provide an alternative approach to detect GBs from the cross-section images more efficiently and accurately. DeCost and Holm applied "bag of visual features" image representation and trained a support vector machine (SVM) for classification of the microstructure dataset (Decost & Holm, 2015). Gupta and Sarkar reported a phase segmentation and GB detection processing technique in plain carbon steel samples using modern image processing operators like linear iterative clustering and skeletonization (Gupta et al., 2019). Friel and Prestridge developed an algorithm to detect GBs accurately by thresholding, followed by a higher-level

✉ John S. Kang
jkang4@sdsu.edu

[1] Department of Mechanical Engineering, San Diego State University, San Diego, CA 92182, USA

Published online: 06 May 2024

boundary completion process to complete the missing sections (Friel et al., 1990).

Deep learning is a process applying a hierarchy of multiple trainable layers used to generalize some objective function relating a set of input and output data (W. J. Zhang et al., 2018). Neural network (NN) based deep learning methods have also come to interest in the past few years. Li and Chen proposed an end-to-end deep neural network (DNN) based on adversary network and feature learning for faster GB detection in Al–Si–Mg alloys (Li et al., 2020). Bordas and Zhang provided an end-to-end workflow of a NN-based holistically nested edge detection (HED) model for autonomous grain size analysis in microscopy images (Bordas et al., 2022).

In addition to the above ML algorithms, some studies have used a combination of different algorithms to improve the accuracy of GB detection. Wei and Peng explored the option of boosting, which is a type of supervised learning algorithm that converts a set of weak learners into a strong one to reduce high bias, followed by principal component analysis (PCA) to extract the location and trace of GBs in a pure aluminum bi-crystal (Wei et al., 2019). Kim and Cho presented a combined NN and fuzzy logic application which uses spatial relationships among neighboring edges around the edge segments of interest to improve GB detection performance in noisy images (J. S. Kim & Cho, 1994).

Even though the current ML methods can be used for image-based GB detection, they have limitations that they require high quality input images with a clear visualization of the GB structure. Clear visualization of GBs often relies on a thoughtful metallographic preparation, which involves employing an appropriate etching methodology based on prior knowledge of the target materials (Rohrer, 2011). For bulk materials, especially those typical materials presented in the previous studies such as carbon steel, these materials already have a mature methodology and a standard operation procedure to follow in material science to help produce clear GB images; however, when it comes to a new material system with lacked prior understanding, the metallographic preparation process needs to start from the common way and go through some trial and error processes to obtain a satisfactory GB structure appearance. This makes the presentation of clear GB to be time-consuming.

Additive manufacturing can be described by the approach of 'design thinking' which is comprised of two functions, creating a solid layer and joining two solid layers together (Tony et al., 2023). Different types of additive manufacturing can then be classified depending on how those two functions are carried out. This study focuses on the binder jetting classification of additive manufacturing, where a powder layer is spread and an adhesive is deposited to bind together selected areas. The iterative layer-wise spreading of powder with subsequent injection of binder forms a solid part. The part is then sintered to remove the binder and consolidate the powder (Mostafaei et al., 2021). Generally, sintering is a manufacturing technology by which powders are consolidated into the strengthened structure with the help of heating, pressure and/or physical fields (Olevsky & Dudina, 2018). It is noted that while fused deposition modeling was initially referred to 3D printing, the term 3D printing has become synonymous for all types of additive manufacturing (Armstrong et al., 2022). Therefore, the terms will be used interchangeably in this study.

In powder-based 3D-printed and sintered materials, the challenge to present clear GBs under the microscope becomes even more pronounced. The presence of anisotropic microstructural features distinguishes the sintering of 3D-printed materials from the isotropic rigid sintering. With this consideration, understanding the microstructural evolution during the anisotropic sintering of 3D-printed materials requires a reliable identification of GBs structure. Such sintering process normally adopts pressure-less (free) sintering to avoid any damages on as-built complex shape; however, in this case, full densification can hardly be achieved. This can bring a challenge to the detection of GBs, as pores may interfere the recognition of GBs and promote the destruction of GBs in etching process, which cause the GBs unclear in the cross-section images as shown in Fig. 1. It is more difficult to clearly image the GBs without introducing other interfering products in newly developed 3D-printed materials by only a few experimental attempts.

Even though cross-sectional GB detection and characterization is critical in sintered 3D-printed materials, little research has been performed on the GB detection and characterization from the cross-section images with unclear GBs and interfering products such as pores on the GB. Therefore, a generalized image-based GB detection and characterization method is needed for sintered 3D printed materials. Traditional image processing methods cannot be applied to the presented task due to the image complexity, but a potential solution is to utilize deep neural networks based on the literature review. However, no studies have been performed that compare various deep learning algorithms for unclear GB characterization. Furthermore, novel materials are difficult and expensive to produce which limits experimental training data sets, so the solution will need to be robust to small datasets. Another challenge with unclear GBs is the predicted GB can be incomplete and thus discontinuous.

To resolve these challenges, in this study commonly used deep neural networks for image segmentation are trained and compared on the GB detection task. These networks include UNet, Ensemble of UNets, Masking ResNetXt, and denoising diffusion masking model (Lai et al., 2023; Ronneberger et al., 2015; Xie et al., 2017). To ensure robustness to small data sets, the presented method adopted image augmentation to increase the training data sets. Morphology functions in

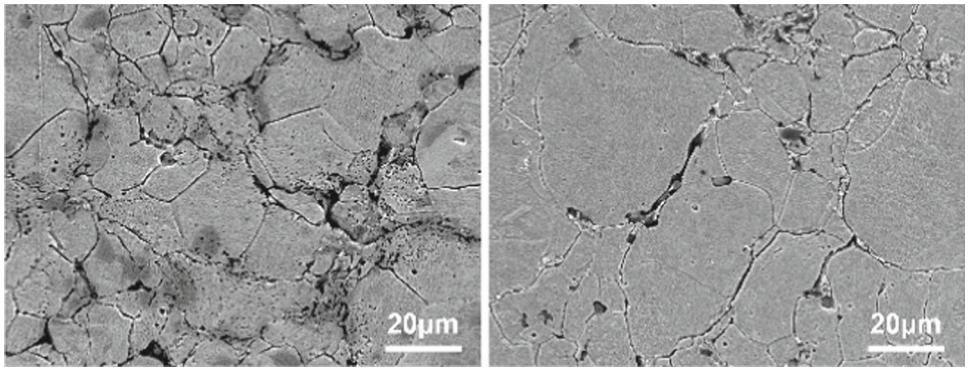


Fig. 1 Exemplary cross-section images of sintered 3D-printed Ni samples

the OpenCV library were used along with denoising diffusion generative networks for the augmentation (Community, 2010; Ho et al., 2020). To prevent discontinuities on predicted GBs, an end of line detection and interpolation between the end of lines algorithms are developed and applied during post processing to connect missing GBs. The proposed method is shown to perform well on the cross-section images with unclear GBs as well as the images with clear GBs. The detailed challenges and methodologies of the proposed method are described in "Approach" Sect. The results of the GB detection and characterization using the proposed methods and the conclusions are provided in "Discussion of the results" and "Conclusion" Sects., respectively.

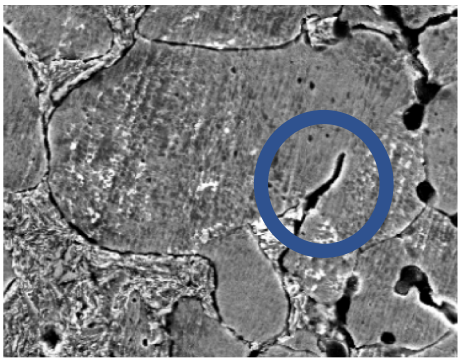


Fig. 2 Example of porous structure with GB features

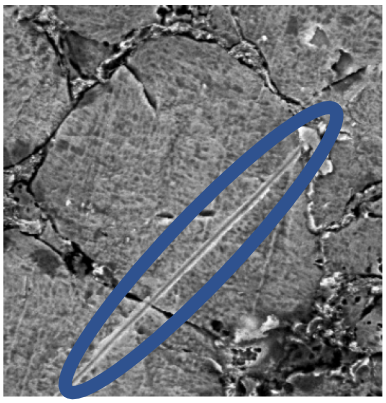


Fig. 3 Surface scratch on etched sample

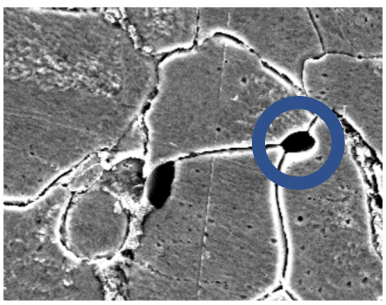


Fig. 4 Example of pore along the GB

Approach

Challenges

To develop a viable method of GB detection and morphological characterization, it is useful to discuss the visual properties of GBs and leverage these in the proposed algorithm. Because the GB exists between two or more grains, it is not possible for a partial GB to exist. Therefore, a GB must propagate continuously, starting and ending at an image boundary. This knowledge can be leveraged to eliminate false GB detection. Figure 2 displays such an instance of a porous structure that resembles a GB. The selected method must be able to discriminate between true and false GBs.

Another example of a possible false GB are surface scratches incurred during sample preparation. These scratches may display light and/or dark features making them easily confused with GBs. However, the scratches are also characterized by lack of curvature and continue across the entire image as shown in Fig. 3. These features can be used to distinguish them from GBs.

Another complication is the existence of pores along the GB as seen in Fig. 4. The pore is a separate feature of interest; however, it is also located along the GB. Therefore, the GB label should not significantly overlap with the pore because

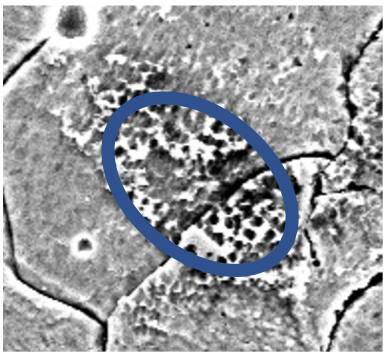


Fig. 5 Etching residue resulting in pore-like structure

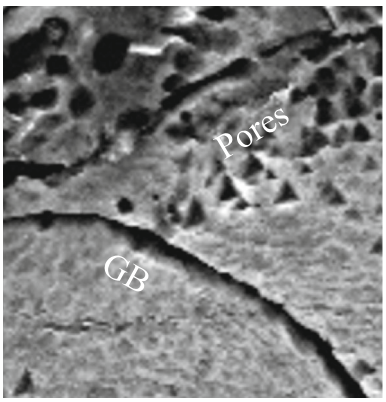


Fig. 6 Comparison of similar features of pores and GB in close proximity

it is desirable to characterize the pores in the future. Large pores lying on GBs must also be distinguishable from grains.

Pores may also be incorrectly identified due to errors. The etching process itself may be a source of confounded porous structure. An example of this residue is presented in Fig. 5. The structure can be easily mistaken for pore clustering without careful consideration by the selected algorithm. It is also noted that running through the cluster is a GB, which adds further complexity to the segmentation requirements.

Another possible source of pore misclassification arises from over-etching of the GB. An example of how similar these two classifications can be is presented in Fig. 6. During etching, the pores in close proximity connect and elongate to form features similar to GBs. Similarly, the etching has eroded the GB such that pores along the GB are barely visible. The selected algorithm must be able to distinguish between these two conditions.

Another challenge with the presented dataset is the appearance of incomplete GBs. These regions represent the biggest challenge in that even among experienced individuals, the existence of the GB may be disputed. These regions are labelled to preserve continuity in GB labels. An example of an incomplete GB is presented in Fig. 7. What appears

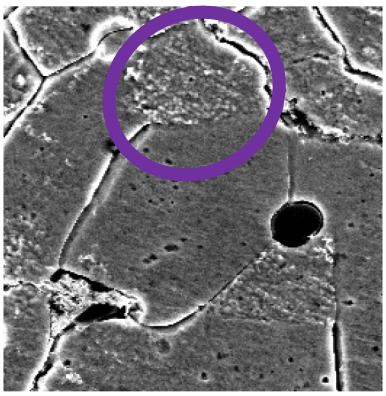


Fig. 7 Example of incomplete GB

to start off as a GB on the bottom left, stops abruptly. Since it is likely the algorithm will label this bottom left region, a choice must be made for the top right region to maintain continuity of the GB.

Various image-based algorithms have been developed to aid process automation. These algorithms can be classified by their intended application and include image classification, object detection, and segmentation. Image classification is the process of labelling an entire image, object detection is used to localize specific objects within an image, and segmentation labels each pixel in an image. Image classification is used when the location of the object within the image is not required. Object detection is used when the location of the object is desired as well as the number of objects. Segmentation is used when the boundary of the object is desired, but the number of objects is not important.

In the presented task of GB detection, the GB is shared by all grains. Therefore, the number of GBs and object detection is of no interest. Furthermore, all images are assumed to contain GBs, so classification is also of no interest. However, which specific pixels represent the GB is of interest. Therefore, image segmentation is the proper image processing method to apply.

The method of image segmentation is also dependant on the task. For images with features containing well defined intensities, thresholding can be applied. Images with well-defined continuous boundaries can be detected with canny edge detection (Canny, 1986), and the resulting edge mask can be filled using a watershed algorithm to segment the image. These ideal scenarios, however, generally require careful image pre-processing and fine experimental control to provide results exceeding a first order approximation. To demonstrate the applicability of classic algorithms a test was performed on a typical image in Fig. 8. Of the algorithms, the threshold filter provides the best results, but it is still inadequate since the resulting mask is very noisy and grain edges are discontinuous. Overall, none of the classic solutions can be used to characterize grains in the presented dataset.

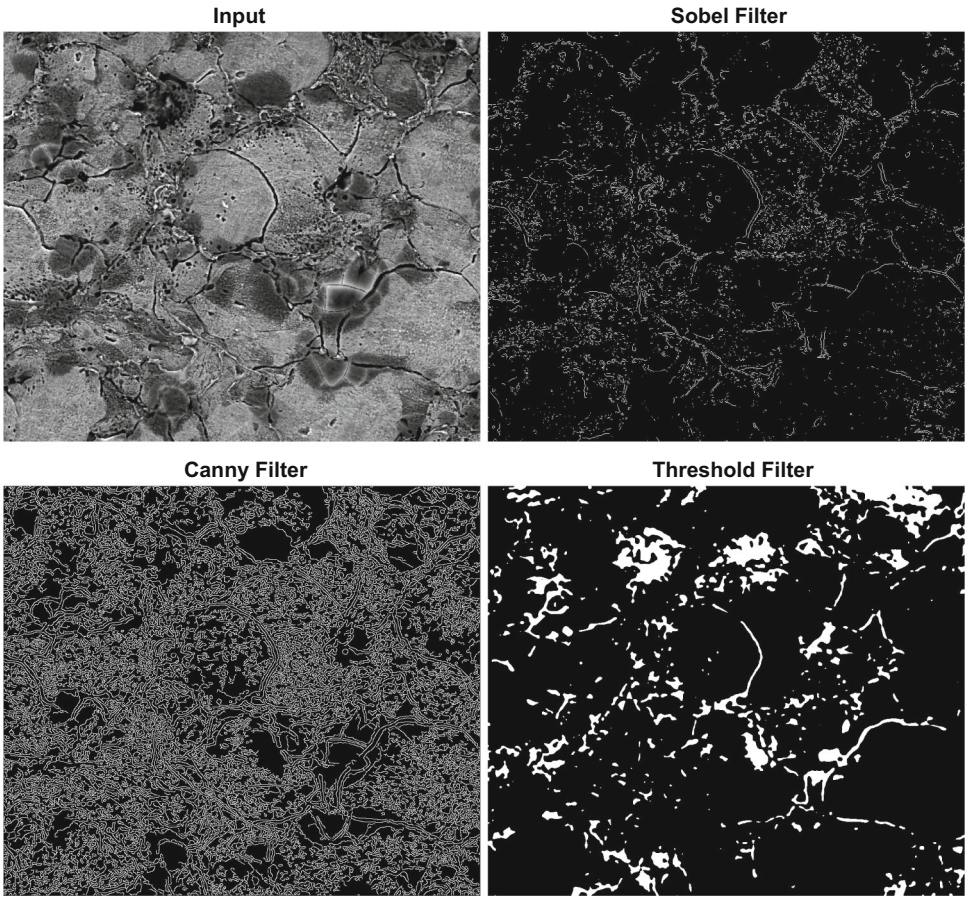


Fig. 8 Classic edge detection algorithms on an etched image with unclear GB

The same edge detection algorithms on an image with clear GBs is shown in Fig. 9 The image was sourced from an earlier grain detection publication (Dengiz et al., 2005). It is shown that the canny filter and simple thresholding perform well for GB detection on the image. This stark difference further illustrates the complexity of the presented GB labelling task with the unclear images.

Due to the complexity of images with unclear grain boundary, manual feature engineering and classic edge detection methods cannot be applied. Therefore, deep neural networks are applied to the problem which perform automated feature selection. However, on small datasets, deep neural networks are capable of overfitting feature selection on the training set resulting in poor performance on other images. To prevent overfitting, the training images set must be increased or augmented. Due to the challenges in collecting additional experimental images in metal 3D printing, image augmentation was performed via a second deep neural network. This network is trained on training image set and the latent space is traversed to generate images that contain features similar to the training set, but sufficiently different to prevent overfitting. The specifics of these techniques are described in the following sections.

Methodology

Given the multiple caveats required in masking GBs properly, a CNN-based deep learning method is preferred. The flexibility of deep learning allows relevant features to be automatically extracted from images and used for classification. Multiple deep learning network architectures are available to perform image segmentation. Some of these will be compared in this study to determine which is most suitable for general grain segmentation.

For each of the models, the output layer is composed of three channels. One channel to label the GB, the second channel to label pores on the GB, and the final channel to label internal pores. Since a given pixel can have more than one label (i.e., a GB and a pore on the GB), a simple mean squared error (MSE) loss function is used following a sigmoid layer.

Instead of directly training the networks at high resolution, a stepping scheme is adopted to speed up training time and increase training variability by initially starting at a lower resolution and increasing. During augmentation, a given image can be cropped in more unique ways with a smaller resolution than a larger one, preventing overfitting. Once the small

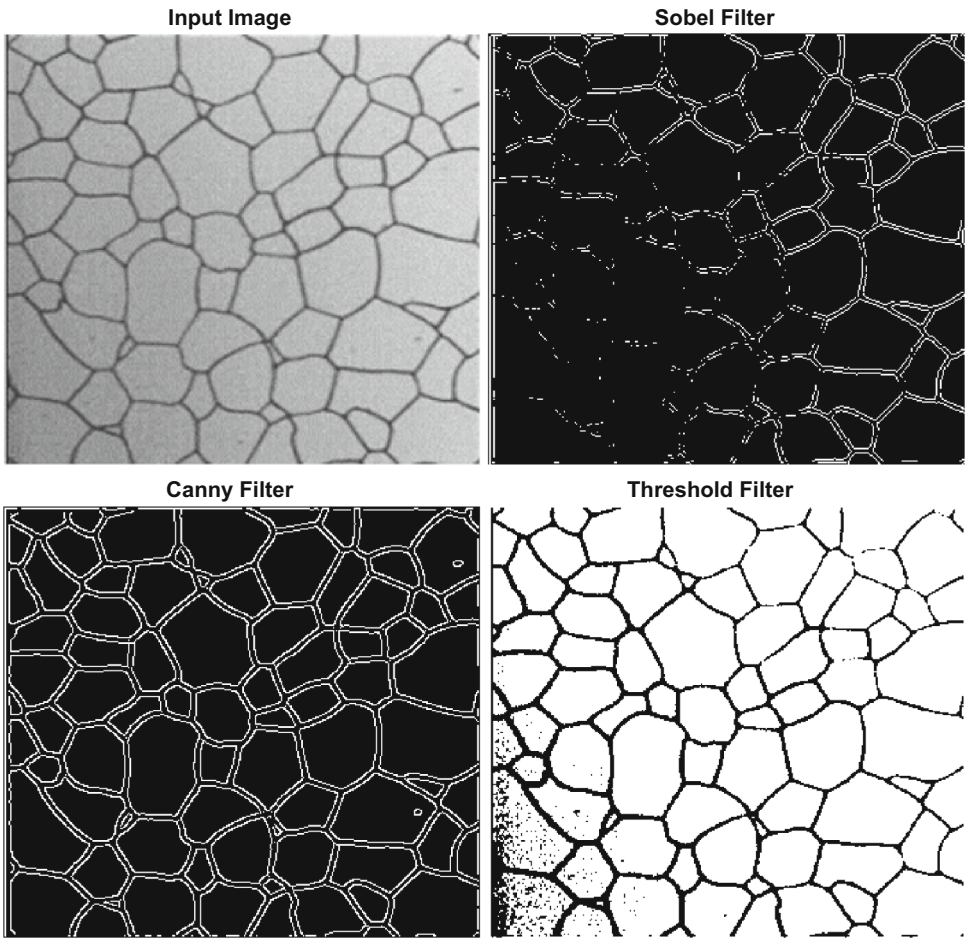


Fig. 9 Classic edge detection algorithms on image with clear GB

Table 1 General training scheme for networks and hyperparameters used during training; the network is initially trained in step 1 then retrained in step 2 and again in step 3 with the specified resolution and hyperparameters to reduce overall training time

Training step	Input resolution	Learning rate	Patience	Initial augmentation	Decay
1	512	0.0001	100	0.25	100
2	640	0.00005	50	0.10	20
3	768	0.00005	50	0.01	20

resolution is trained, the network has approached the minimum, to remain in this vicinity, the learning rate is reduced before subsequent transfer learning at higher resolution.

The training scheme is presented in Table 1. The networks were trained with three steps. The initial training step utilized the lowest resolution, highest learning rate, and highest probability for augmentation. The network is trained with patience of 100 epochs on the validation set (i.e., if the validation loss does not decrease for 100 consecutive epochs, the training stops). The augmentation policy is handcrafted and consists of two variables, initial augmentation and decay. The augmentation probability was set at 25% with a decay of 100 epochs. After the networks is completes training at 512 $\times$ 512 cropped image resolution, the resolution is increased and the model is fine-tuned with parameters listed for subsequent steps. The training scheme presented in Table 1 was determined by performing grid search on the learning rate, patience, augmentation and decay with a UNet. Due to the long time required to perform the grid search, these parameters were also used on the other deep neural networks for GB segmentation.

In addition, generative deep neural network models are utilized to augment the image set and prevent overfitting. In this study we compare generative adversarial networks (GAN) with denoising diffusion networks for augmenting the training set. However, these models are also susceptible to overfitting, so classic image augmentation is applied during

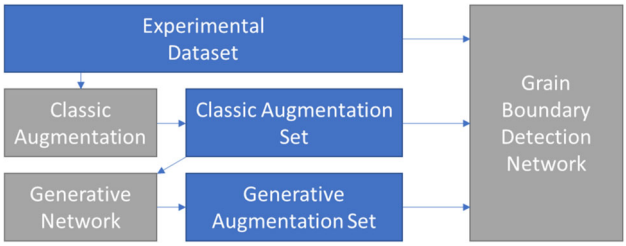


Fig. 10 Overall training flowchart for GB detection

training to vary the input images. The training flowchart is displayed in Fig. 10.

U-NET

The UNet architecture is commonly used for applications of image segmentation. The structure allows for features of the input image to be extracted in the convolutional layers. Then the mask is generated in the from the extracted features by the transpose convolutional (up-conv) layer outputs (Ronneberger et al., 2015). At each layer, extracted features are copied to the outputs as shown in Fig. 11. The resulting segmented image can capture fine features from the input image because features are extracted and copied to the output at each level.

In the original UNet implementation, the input size (572×572) is larger than the output mask (388×388) (Ronneberger et al., 2015). This presents an issue for the presented application because the grain boundaries are thin and pores are small, so downscaling the mask may result in loss of regions. The presented implementation differs from the original architecture by increasing the stride of the transpose convolutional layers so that the layers dimensions copied from the encoder side of the UNet match the decoder size without cropping. The resulting mask is then the same size as the input image and small features can be resolved.

In addition, the loss function of the UNet is modified from the original implementation from cross entropy loss to mean squared error loss. This is because a grain and a pore can exist in the same location, and cross entropy loss assumes that each pixel is a unique class. Therefore, to use cross entropy loss an additional class would need to be provided for this case. Alternatively, mean squared error loss can used without this requiring additional classes.

The UNet hyperparameters consist of the number of channels at each level. In Fig. 11, there are 64, 128, 256, 512, and 1024 channels. In this study, the number of convolutional layers is varied to 32, 64, 128, 256, and 512 channels as well test performance as a function of complexity. In addition, an ensemble of UNets is trained on the presented task with pseudocode presented in Fig. 12. A framework of ensemble inference has been conceived previously (Modi et al., 2011).

Within the context of framework, each of the UNets trained are experts in their respective tasks and respond to different cues within the data. The results of each UNet are then combined to produce a single prediction. The ensemble consists of four UNets, one to predict the GB mask, one to predict the inverted GB mask, and two to predict the residuals of each mask. The first two UNets share two loss functions (Eqs. 1 and 2).

$$L_{mult} = (Pred_1 \times Pred_2)^2 \tag{1}$$

$$L_{add} = (Pred_1 + Pred_2 - 1)^2 \tag{2}$$

The inverted mask should be 0 everywhere the non-inverted mask is 1. Therefore Eqs. 1 and 2 enforce these properties. The other two networks simply predict the residuals of the first two and are trained with the following loss function (Eq. 3) after the first two networks are trained.

$$L_{res} = \left(Pred_{res,1} - max\left(min\left(Pred_1 - Target, 0\right), 1\right)\right)^2 \tag{3}$$

The min and max function are represented by the clamp function in pytorch. The implementation is used to only include the positive residuals, or simply, the masks that are not predicted by the network.

The motivation for training a network on the positive residuals only is so that network performs line completion and reduce grain discontinuities; that is the residuals networks is a specialist for line completion because it is trained to only predict areas missed by the first network. The motivation for predicting the inverted mask (with the inverted image as input) is for the second network to predict grains opposed to GBs. It is anticipated that different features are required for this task and may improve generalization. During testing, the networks are averaged with their residuals to yield the output using Eq. 4.

$$Pred_{ens} = \frac{\left(Pred_1 + Pred_{res,1}\right)}{2} + \frac{1 - \left(Pred_2 + Pred_{res,2}\right)}{2} \tag{4}$$

An example of the UNet ensemble prediction is presented in Fig. 13. The residuals tend to weight areas that are weakly predicted by the primary network, resulting in line completion. However, the residuals can also incorrectly predict grain boundaries. Therefore, residuals can aid in line completion and prevent incomplete grain boundaries, but they do so at the possibility of incorrect grain boundary prediction.

In total three UNets were developed for the presented task. One UNet with 32 initial layers, one with 64 initial layers,

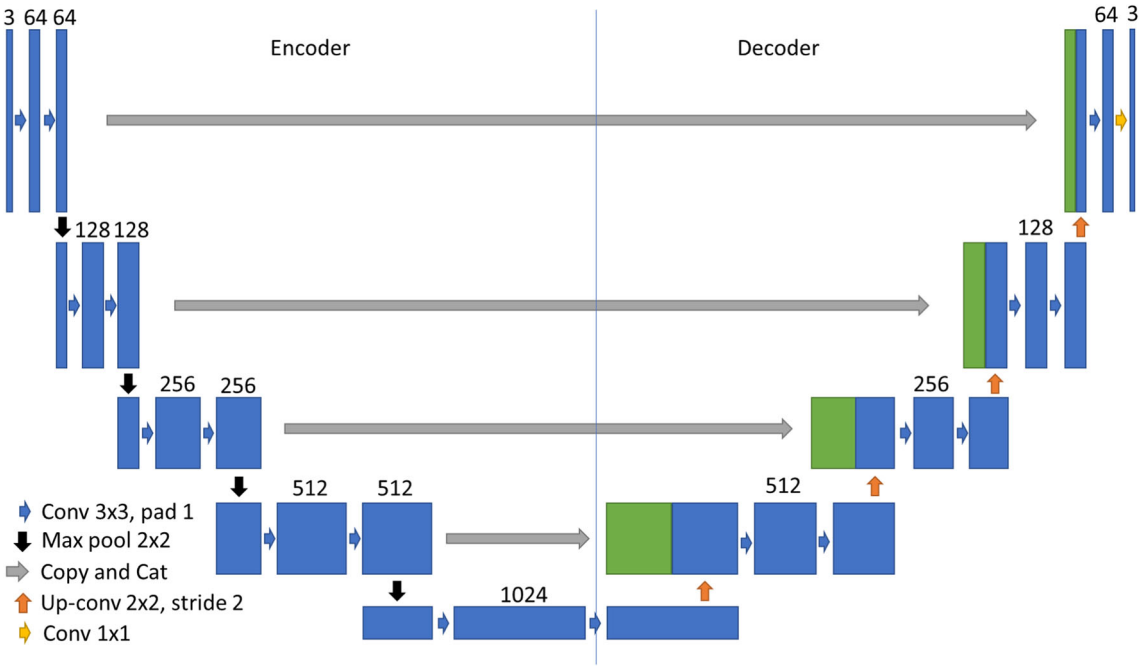


Fig. 11 UNet architecture displaying the feature extraction convolutional layer and the generation up-convolution layers (Ronneberger et al., 2015)

Fig. 12 The pseudocode for the UNet ensemble for the training and prediction

```
# Training UNet Ensemble
for input, mask in train_set:
   pred1  = UNet1(input)
   pred_res1 = UNet2(input)
   pred2  = UNet3(~input)
   pred_res2 = UNet4(~input)
   loss1 = (pred1 - mask)**2
   loss2 = (pred2 - mask)**2
   lmult = (pred1 * pred2)**2
   ladd = (pred1 + pred2 -1)**2
   lres1 = (pred_res1 -clamp(pred1,0,1))**2
   lres2 = (pred_res2 -clamp(pred2,0,1))**2
   loss = loss1 + loss2 + lmult + ladd + lres1 + lres2
   loss.backward()

# UNet Ensemble Prediction
for input, mask in val_set:
   pred1  = UNet1(input)
   pred_res1 = UNet2(input)
   pred2  = UNet3(~input)
   pred_res2 = UNet4(~input)
   mask_out = (pred1 + pred_res1)/2 + (1-(pred2 + pred_res2))/2
```

and the ensemble with 32 initial layers. Each successive layer is two times the previous with a total of 5 encoding layers and 4 decoding layers as shown in Fig. 11.

ResNeXt

ResNeXt is a deep neural network like ResNet but differs in that the building blocks of each layer are repeated in parallel, which provides a new dimension known as cardinality, or simply put, the width of the network (Xie et al., 2017). The key takeaway of this design is enhanced accuracy without increasing complexity as compared to the ResNet architecture (Xie et al., 2017). We adopt the ResNeXt model to determine if the increased network complexity (with respect to the UNet) can further generalize the presented task.

The model was modified to include a Unet-like head where layer outputs of the ResNeXt model body are concatenated to successive transpose convolutions of the Unet head. The

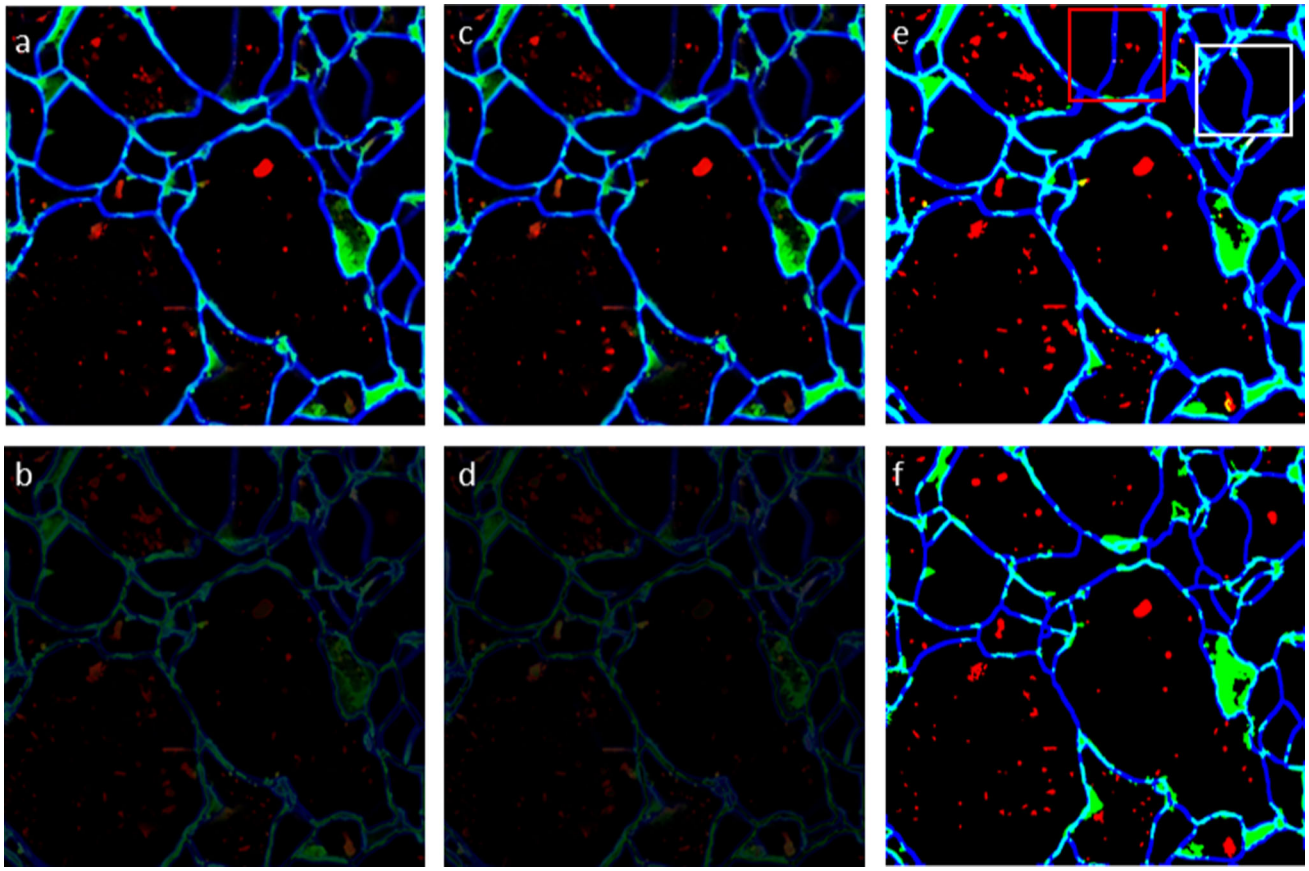


Fig. 13 Output of each network from the UNet ensemble, **a** prediction 1, **b** prediction 1 residual, **c** prediction 2, **d** prediction 2 residual, **e** ensemble output, and **f** ground truth output. The white box indicates an area that correctly predicted due to residuals and the red box indicates an area that is incorrectly predicted due to the residual networks

four middle layers of the ResNeXt are utilized since the dimensions are scaled by 2 for each output and can be easily matched to the transpose convolution output layers. The final two output layers are used to upscale the dimensions to their original size and subsequently reduce to the desired number of channels. The sigmoid function is applied to the final output to perform pixel-level classification. The pseudocode for adapting a UNet heat to ResNeXt model is presented in Fig. 14 and a diagram for the implemented architecture is presented in Fig. 15.

The pretrained ResNeXt network was utilized with the initial layer modified so the input is a single channel to match our image set. In addition, the final fully connected layer and classification layer of the ResNeXt is removed. The input and output layers for the Unet head were selected based on the dimensions shown in Fig. 15. The transpose convolutional layers have the same number of input and output channels, and the subsequent convolutional layers reduce the channel size by half in the first convolution as shown in Fig. 15.

The pretrained ResNeXt50 model was tested as the model body. The model uses 32 convolutional groups with 4 input and output channels. The model was obtained from the official pytorch repository. Though higher complexity models can also be tested with the proposed architecture, they were not included in this study due to the increased training time required.

Denoising diffusion segmentation

Denoising diffusion networks are generative models capable of creating realistic images (Ho et al., 2020). These models are trained by iteratively adding gaussian noise to an input image. Then the parameters of a Markov process are learned by the network to uncover the original input image. Images can be labelled during training by including an embedded layer that corresponds to a classification. During image generation the embedding layer guides the network to generate images with features common to that classification.

To perform segmentation, gaussian noise is applied to the mask and the input image is used as the embedding layer as shown in Fig. 16. The denoising architecture is based on the UNet. The UNet implemented for the diffusion network

```
# pretrained resnext50 model
resnext = resnext50_32x4d(weights=ResNeXt50_32X4D_Weights.IMAGENET1K_V1)
# copy the weights into a single channel
temp = resnext.conv1.weight[:,0:1,:,:].clone()
new_layer = nn.Conv2d(1, 64, kernel_size=7, stride=2, padding=3,bias=False)
new_layer.weight[:,:,:,:].data[...] =  Variable(temp, requires_grad=True)
my_model = nn.Sequential(*list(resnext.children())[:-2])

class ResNextUnet():
    def __init__(self, il = 32):
        super().__init__()
        # Define number of channels for each layer
        l = [il, il*2, il*4,il*8,il*16]
        # Define all resnext layers with skip connections
        self.output_layers = [0,1,2,3,4,5,6,7]
        self.k = ['0', '1','2', '3', '4', '5', '6', '7']
        self.pretrained = my_model
        self.selected_out = OrderedDict()
        self.fhooks = []
        # Define all upconv layers
        self.u0 =  self.up_sample(2048, l[4],2)
        self.ud0 = self.down_sample(l[4], l[4], 3)
        self.u1 =  self.up_sample(l[4],l[3],2)
        self.ud1 = self.down_sample(l[3], l[3], 3)
        self.u2 =  self.up_sample(l[3], l[2],2)
        self.ud2 = self.down_sample(l[2], l[2], 3)
        self.u3 =  self.up_sample(l[2], l[1],2)
        self.ud3 = self.down_sample(l[2], l[1], 3)
        self.u4 =  self.up_sample(l[1], l[0],2)
        self.ud4 = self.down_sample(l[0], l[0], 3)
        self.out = nn.Conv2d(l[0], 3, kernel_size=1)
        self.cl = nn.Sigmoid()
        # Store all resnext output layers into list
        for i,l in enumerate(list(self.pretrained._modules.keys())):
            if i in self.output_layers:
                self.fhooks.append(
                 getattr(self.pretrained,l).register_forward_hook(self.forward_hook(l)))
    # Downsample block extracts features to upsample
    def down_sample(self,ci, co, k, s=1, d=1, p=1):
        block = nn.Sequential(nn.Conv2d(ci, co, kernel_size=k, stride=s, dilation=d, padding=p),
        nn.BatchNorm2d(co),nn.ReLU(inplace=True),
        nn.Conv2d(co, co, kernel_size=k, stride=s, dilation=d, padding=p),
        nn.BatchNorm2d(co), nn.ReLU(inplace=True))
        return block
    # Upsample block increases the size to desired width and height
    def up_sample(self,ci, co, k, s=2, d=1, p=0):
       block = nn.Sequential(nn.ConvTranspose2d(ci, co, kernel_size=k, stride=s, dilation=d, padding=p),
       nn.BatchNorm2d(co),nn.ReLU(inplace=True))
       return block
    # Used to access resnext outputs as a hook
    def forward_hook(self,layer_name):
        def hook(module, input, output):
            self.selected_out[layer_name] = output
        return hook
    # Defines skip connects from resnext to unet head
    def unet(self,x,t="u"):
        xo = self.pretrained(x)
        y0 = self.u0(xo)
        out6 = self.selected_out['6']
        y0b = self.ud0(out6+y0)
        y1 = self.u1(y0b)
        out5 = self.selected_out['5']
        y1b = self.ud1(out5+y1)
        y2 = self.u2(y1b)
        out3 = self.selected_out['3']
        y2b = self.ud2(out3+y2)
        y3 = self.u3(y2b)
        out2 = self.selected_out['2']
        y3b = self.ud3(out2+y3)
        y4 = self.u4(y3b)
        y4b = self.ud4(y4)
        out =self.out(y4b)
        return out

    def forward(self, x):
        x1 = self.unet(x)
        out = self.cl(x1)
        return out

inti __main__()
  net = ResNextUnet()
  net.forward(torch.rand(1,3,640,640))
```

Fig. 14 Python code for attaching a UNet head to a pretrained ResNeXt model with architecture shown in Fig. 15. An example of a random image inference through the network is shown in the main function

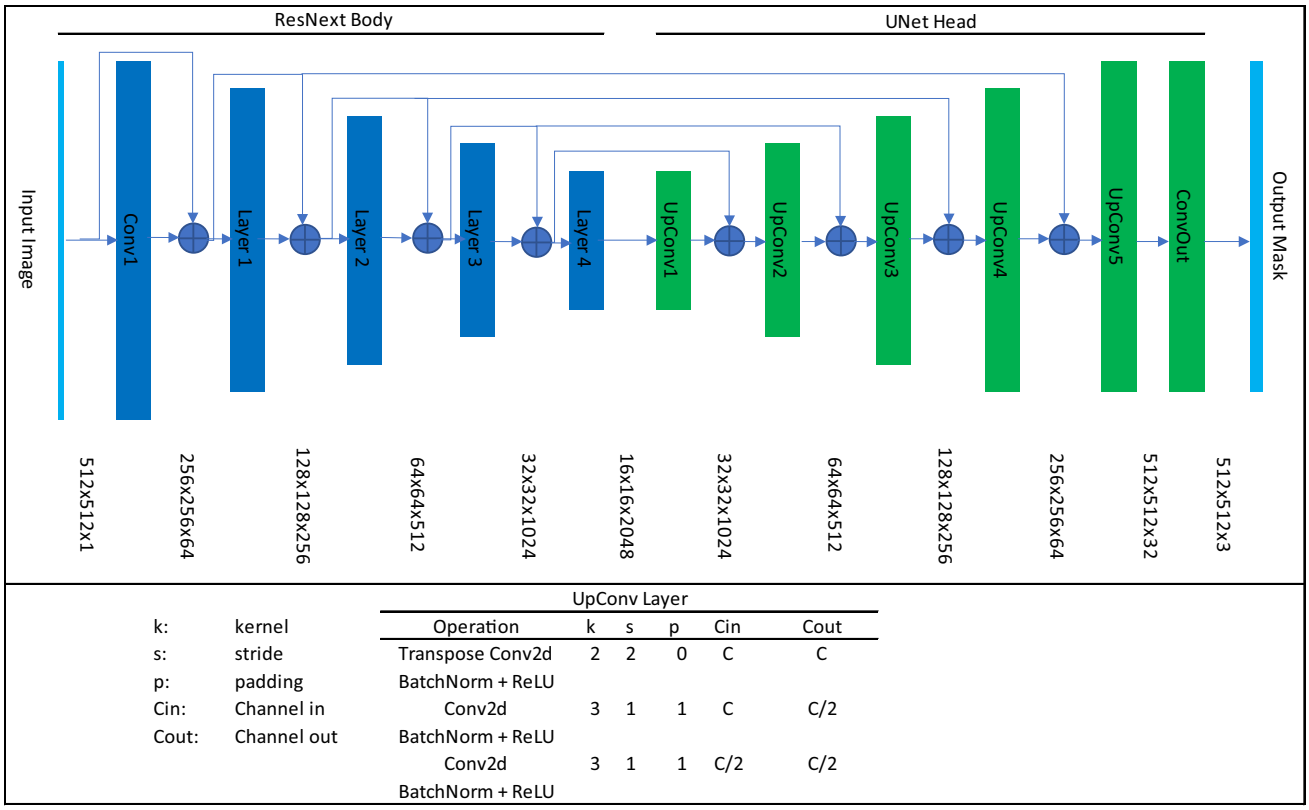


Fig. 15 Modified ResNext architecture for image segmentation including tensor shape at each layer output and architecture for each upscaling layer (UpConv)

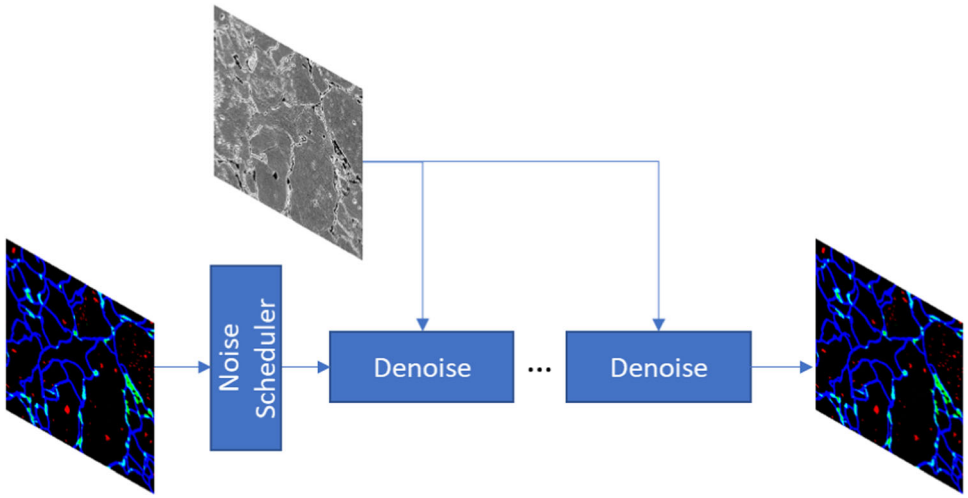


Fig. 16 Denoising diffusion mask generation guided by embedded input image during denoising. Following training, the noise scheduler is removed and only the input image and random noise is used to generate the mask

utilized 32, 64, 128, 256, and 512 channels. When noise is passed through the UNet structure it is decoded a single step and begins to resemble the mask. The number of steps iterated is a hyperparameter and impacts the final quality of mask. Too few steps result in a noisy image, but too many steps require excessive time and may overfit the mask to the image.

To track which iteration the diffusion network is processing, time embeddings are utilized. Time embeddings shift and scale the noisy image to inform the network. The image embeddings can be applied in a similar way to the same step to guide the network to the correct mask. A 2D convolution block with a 1×1 kernel is used to match the correct number of channels and interpolation is used to scale the image to the correct height and width. The relevant code is shown in Fig. 17. After the model is trained, masks are generated from random noise and the embedded input image. Therefore, the

```
class Block(nn.Module):
    def __init__(self, dim, dim_out, groups = 8):
        super().__init__()
        self.proj = WeightStandardizedConv2d(dim, dim_out, 3, padding = 1)
        self.norm = nn.GroupNorm(groups, dim_out)
        self.act = nn.SiLU()

    def forward(self, x, scale_shift = None):
        x = self.proj(x)
        x = self.norm(x)

        if exists(scale_shift):
            scale, shift = scale_shift
            x = x * (scale + 1) + shift

        x = self.act(x)
        return x

class ResnetBlock(nn.Module):
    def __init__(self, dim, dim_out, *, time_emb_dim = None, classes_emb_dim = None, groups = 8):
        super().__init__()
        self.mlp = nn.Sequential(
            nn.SiLU(),
            nn.Linear(int(time_emb_dim), dim_out * 2)
        ) if exists(time_emb_dim) or exists(classes_emb_dim) else None

        self.block1 = Block(dim, dim_out, groups = groups)
        self.block2 = Block(dim_out, dim_out, groups = groups)
        self.res_conv = nn.Conv2d(dim, dim_out, 1) if dim != dim_out else nn.Identity()
        self.res_conv2 = nn.Sequential(
            nn.SiLU(),
            nn.Conv2d(3, dim_out*2, 1))

    def forward(self, x, time_emb = None, img_emb = None):
        scale_shift = None
        if exists(self.mlp) and exists(time_emb):
            cond_emb = time_emb
            cond_emb = self.mlp(cond_emb)
            cond_emb = rearrange(cond_emb, 'b c -> b c 1 1')
            scale_shift = cond_emb.chunk(2, dim = 1)
        xo = torch.nn.functional.interpolate(img_emb,x.shape[2], mode='bilinear')
        h = self.block1(x, scale_shift = scale_shift)
        img_shift_scale = self.res_conv2(xo).chunk(2, dim = 1)
        h = self.block2(h, scale_shift = img_shift_scale)
        return h + self.res_conv(x)
```

Fig. 17 Code to add image embeddings to the residual blocks for masking application. Code is modified directly from source in (Ho et al., 2020)

diffusion network learns how to generate image masks guided by the embedded input image.

Training the denoising diffusion segmentation network required significant time. Compared to the denoising diffusion generative network discussed in "Deep network image augmentation" Sect., the denoising diffusion segmentation network required about 15 times as long to train (approximately two weeks) on 512 × 512 images. As a result, the network was not further trained at higher resolutions. Therefore, additional training may further improve the network performance but would require approximately an additional month.

Classic image augmentation

Classic image augmentation is the process modifying an image with geometric transformation and photometric shifting techniques (Khalifa et al., 2022). Applying these techniques to the image set increases image variability in a controllable and computationally inexpensive manner. Augmented images remain accurate because source images are from the training set and remain within the task domain. Furthermore, operations performed are repeatable, so geometric transformations performed on the images can be repeated on the masks to maintain consistency. Due to the diversity of the image set and its relatively small size, classic image augmentation is essential for reducing overfitting when training deep neural networks.

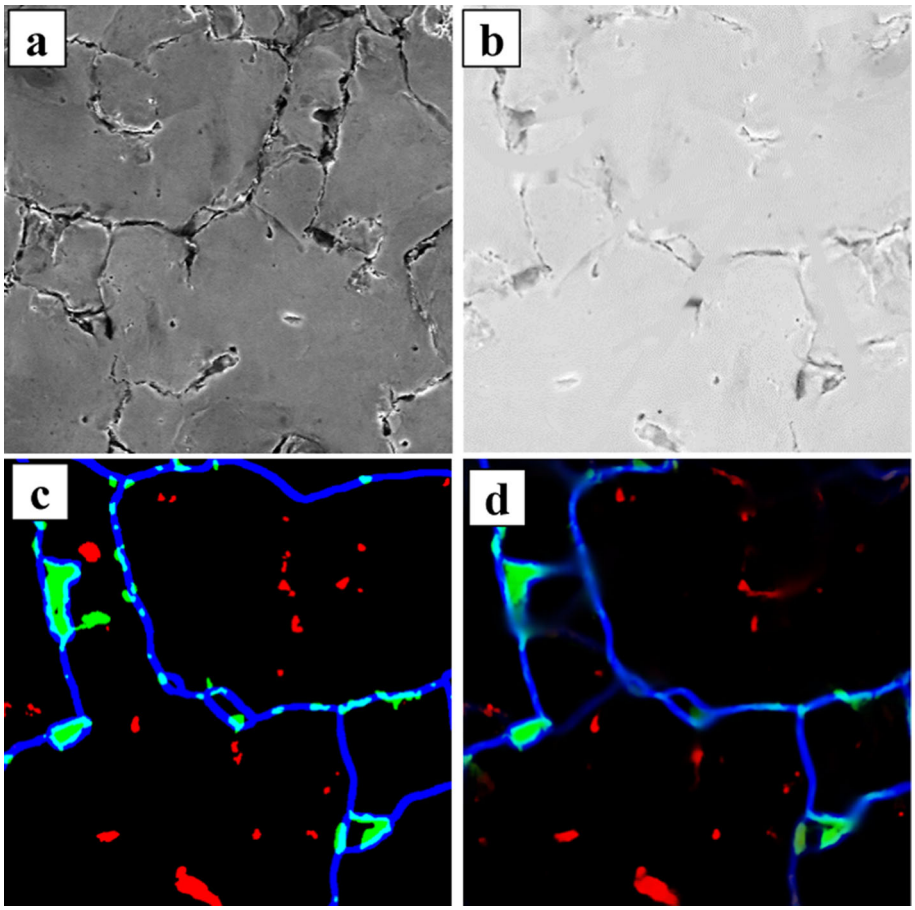


Fig. 18 **a** Input image, **b** augmented image, **c** ground truth mask, and **d** mask prediction

Augmentation performed includes image scaling, random cropping, random gamma, image distortion, and random noise addition. To further increase image variability, each function is called in random order and each can be weighted independently or combined. In addition, we introduce a healing mask implemented in OpenCV, where a background-colored line is drawn to obfuscate the image, then is subsequently healed via the OpenCV inpaint algorithm (Community, 2010). The implementation simulates an unclear GB and forces the neural network to perform interpolation over the boundary. An example of an augmented image with cropping, mirroring, gamma adjustment, and healing is provided in Fig. 18 along with the ground truth mask and the predicted mask during training. The neural network reasonably interpolates between the obfuscated areas. However, areas that are too obfuscated are not interpolated (e.g. the top right of the augmented image). This augmentation increases the loss when predicted GBs are discontinuous during training.

During training each augmentation algorithm was initially set to occur with a probability, p_o, and decrease every iteration via a half-life, λ, via the decay function (Eq. 5).

$$p_{augment} = p_o \exp\left(-\frac{\ln(2)}{\lambda} epoch\right) \tag{5}$$

The hyperparameters used during training are reported in Table 1. In general, it was found that initially increasing the half-life parameter improves overall model performance, but increasing it excessively results in over-regularization and further training time is required. Decreasing the half-life parameter tended to result in overfitting where conversion occurred quickly but at a decreased performance. The initial probability parameter is set to ensure high variability is achieved while keeping augmented images within the task domain. If the value is set too high, the image is modified excessively and no longer resembles the source images. Values were selected by performing a simple grid search and increasing λ by 20 epochs and p_o by 5%.

Deep network image augmentation

In addition to classic image augmentation, deep learning networks, were used to generate an augmented image set. Two different types of deep generative networks, StyleGAN3

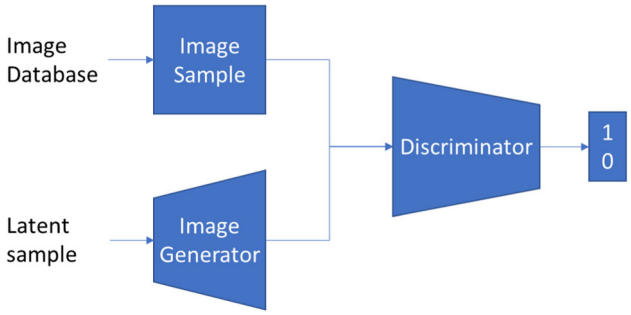


Fig. 19 GAN architecture overview consisting of generator and discriminator, with the discriminator correctly identifying the real and fake samples

and denoising diffusion, were selected for image augmentation. The StyleGAN3 network generates images based on an adversarial approach. Two networks, a generator and discriminator are trained simultaneously as shown in Fig. 19. The generator's goal is to produce images that fool the discriminator, and the discriminator's job is to determine if an image is real or fake. Therefore, as the networks train and each becomes better the respective task, more realistic images are generated. When training is complete, the image generator is used independently to generate images.

The denoising diffusion model is of the same design discussed in Sect. "Denoising diffusion segmentation" but does not contain the embedding images. This is because there is no need to guide the diffusion within a specific latent space, it is only necessary to generate images that are similar to the input image set.

The StyleGAN3 model and diffusion model were train on the data and compared at low resolution (Ho et al., 2020; Karras et al., 2021). The results of this initial study showed similar quality between the generative models as shown in Fig. 20 However, when attempting to generate larger 640 $\times$ 640 images, it became clear that the diffusion model was advantageous in that training only required about a day, while the StyleGAN3 model required weeks to train. As a result, the diffusion model was selected for image augmentation. The diffusion model was only trained on the training image set to maintain independence from testing images. The final images were trained at 800 $\times$ 800 resolution.

Post processing

The resulting GB masks are not guaranteed to be continuous, that is, gaps in the predicted GB can occur despite the augmentation performed during training. For the predicted grain morphology to be accurate, these boundaries require post processing to remove discontinuities. This is accomplished by interpolating the end of predicted GB line to connect with the closest intersecting GB. In addition, the GB width is on the atomic scale and therefore for accurate grain morphology, the predicted boundary must be skeletonized so as not to reduce the grain size. An example of GB post processing is shown in Fig. 21.

To perform the post processing the GB channel is isolated from the mask. The GB is scaled to 512 $\times$ 512 pixels so that consistency between image post processing is maintained. Next a dilation and subsequent erosion of the mask is performed with a 4 $\times$ 4 kernel. This stage connects GBs that are close together, i.e., parallel GBs and those that are not quite connected.

It is then skeletonized using OpenCV and all non-zero pixels are set to one (T. Y. Zhang & Suen, 1984). Using the skeleton image, end of lines locations can be easily determined by padding the image by two pixel and performing convolution over the image with a 3 $\times$ 3 kernel. The convolution output is then multiplied by the skeleton input to only include location of the input image. The end of the lines are then all pixels with a value of two, indicating that only itself and a single neighbour are present at the location. Finally, the border pixels are removed, and the final output is obtained. Once the end of the line is determined, the line is traced back 15 pixels or until a branch is detected which are used to extend the line via linear interpolation until it intersects with a GB or image border. The progression is shown in Fig. 22.

Each line is stored in a list and trimming is performed if intersection between interpolated lines has occurred. It is possible that interpolated lines may intersect more than once and trimming results in a discontinuous line. To prevent this case the end of line algorithm is performed a second time. During testing it was found that subsequent end of line interpolations tends to result in incorrect GBs, therefore, any discontinuous lines after the first two iterations are simply discarded. The interpolation, while not fully optimized, is time consuming, taking an average of 120ms on a 512 $\times$ 512 image. For comparison, the inference time of the UNet model takes an average of 20ms. Therefore, the inclusion of postprocessing makes it impractical for video applications, but acceptable for processing static images.

After GBs are determined, each grain is masked by selecting a random unmasked pixel and performing a floodfill algorithm as shown in Fig. 23. Once all areas are masked, the grain masks are compared to the porosity masks. Occasionally, pores on the GB are large enough to be marked as a grain. If the IOU between a pore and a grain is greater than 0.5, the grain is removed from the list and assumed to be a pore. An example of pore removal is show in Fig. 24. The resulting grain population is then used to compute the grain morphological properties of the image.

The grain properties computed include the grain area, solidity, aspect ratio, extent, equivalent diameter, angle, and average curvature. The definition and an example of each of

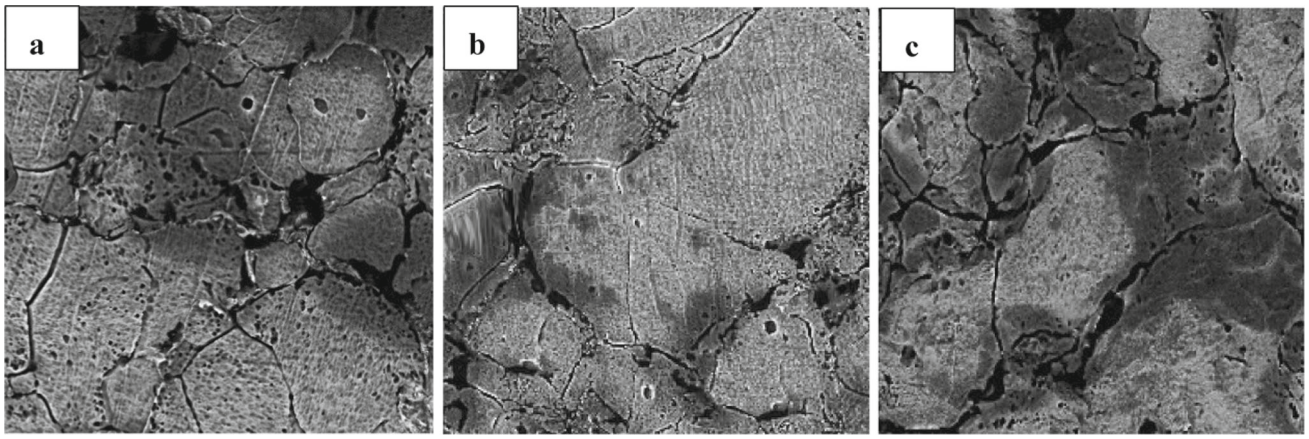


Fig. 20 Comparison of **a** real image, **b** StyleGAN3 image, and **c** Diffusion image

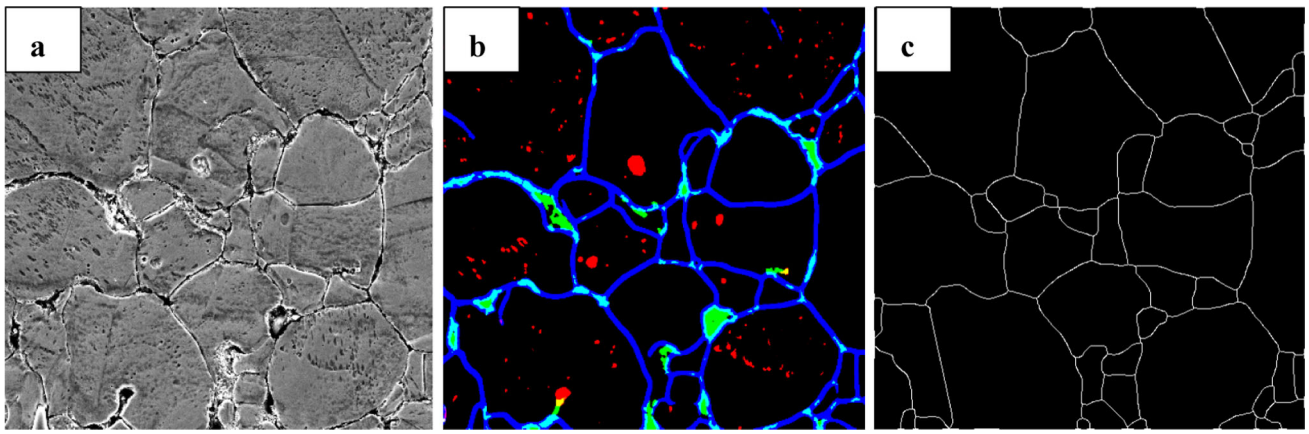


Fig. 21 **a** Input image, **b** predicted mask, **c** and GB post processing

those properties are listed in Table 2. Each of the grain properties is averaged and compared to the ground truth to be used as an accuracy metric. In addition, the standard deviation of the grain size as well as the number of grains is recorded and used in the metric. The mean average of each property is then computed and reported for the final evaluation.

Data preparation

Experiment setup

The commercial gas atomized pure Ni powders (99.95% purity and ~ 20μm in average size) were used as the raw materials for the sample preparation by binder jetting technology (Fig. 25a). Sugar and maltodextrin with the same mass fraction were used as the binder making up 1wt.% of the mixture. The printing powders were mixed in a conventional dry mixer (Turbula®, WAB-Group, Switzerland) for 2 h. The cubic Ni samples with the dimension of 10mm × 10mm × 10mm were printed in a custom-made binder jetting printer with a layer thickness of 150μm. Then the printed Ni cubes were subjected to the curing process at 60 °C for 24 h to bind the particles and stabilize the printed structure.

Before sintering process, the debinding process was conducted in the tube vacuum furnace (GSL-1700X, MTI Corp., USA) at 600 °C for 30 min. The temperature profile used for de-binding process is shown in Fig. 25b. A slow heating rate (2 K/min) with multiple isothermal stage is to prevent the collapse of printed structures due to the rapid vaporization of binder. Later, the printed Ni samples were consolidated by pressure-less sintering in the tube vacuum furnace (GSL-1700X, MTI Corp., USA) with a heating rate of 5 K/min and a temperature of 1400 °C, as shown in Fig. 25c In this work, four different dwelling time of 0, 30, 60 and 90 min were applied to investigate the efficiency of ML to detect the GBs of 3D printed samples with different densifications. The relative densities of each sintered specimens were measured using the Archimedes method.

The sintered cubic Ni samples were cut along the building direction to obtain their corresponding cross-section images. The preparation procedures were following the regular metallographic method, including the grinding of their surfaces

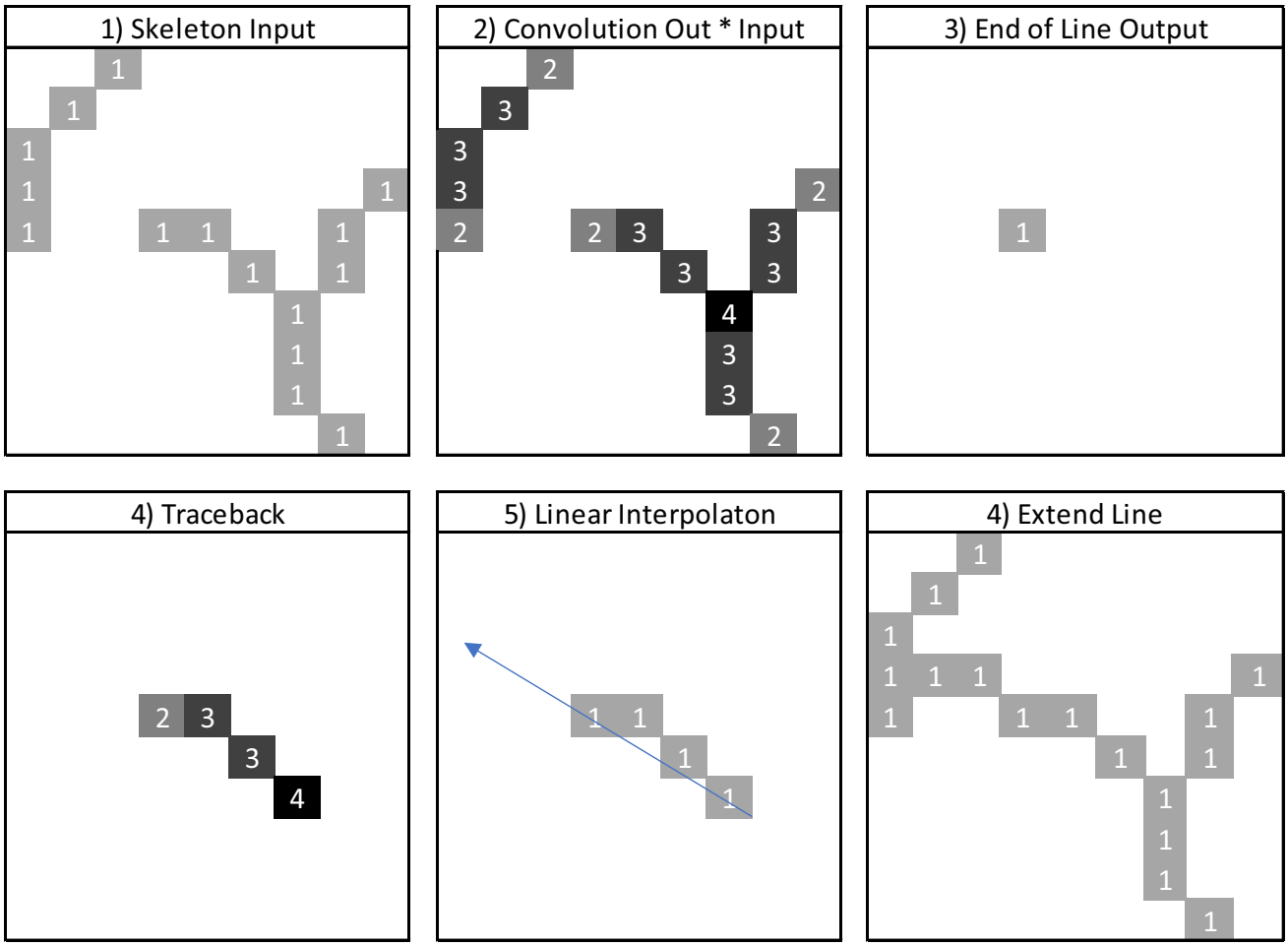


Fig. 22 Progression of end of line detection algorithm after skeletonization and subsequent line interpolation

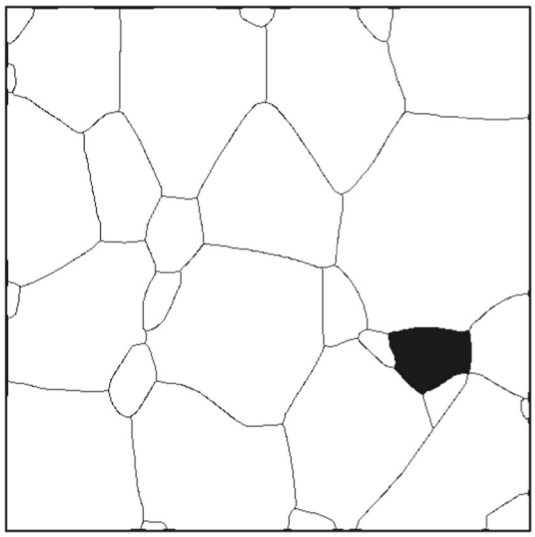


Fig. 23 Grain selected from GB image using floodfill

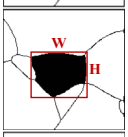
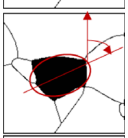
by sandpapers up to 4000 grit and the following polishing by 1 μm diamond suspension and OP-S oxide suspension (Struers, Denmark). To make the GBs visible, the etching of cross-section area of cubic Ni samples was conducted by immersing them in HNO_3/CH_3COOH aqueous solution for 5 s. Finally, scanning electron microscope (SEM, FEI Quanta 450, Thermo Fisher Scientific, USA) was used to obtain the images of cross-section microstructure of all 3D printed Ni samples. Here 150 images for each sample were collected as the data set for the ML algorithm development. Some exemplary SEM images with GBs and unexpected disturbing noises are shown in Fig. 26.

Ground truth data generation

GBs of each image were manually traced by an experienced person to generate the ground truth dataset. An example image with GB mask is presented in Fig. 27. After manual tracing was performed, the masks were skeletonized and then dilation with a 5 × 5 ellipse kernel was performed to ensure that the mask GB mask thickness is unbiased.

Pore masking was not performed manually. It is estimated that over 150,000 pores are contained within the

Table 2 The grain properties characterized by the proposed method

Property	Definition	Example image
Grain area	Area of the selected region in (pixels^2)	
Solidity	The area fraction of the selected region to the convex hull region (blue and black region) $\frac{Area_{grain}}{Area_{convex}}$	
Aspect ratio	Ratio of the width to height of the selected region $\frac{W}{H}$	
Extent	Ratio of the pixels in the selected region to those of the bounding box $\frac{Area_{grain}}{Area_{box}}$	
Equivalent diameter	The diameter of a circle with equal area to the selected region (pixels)	
Angle	Angle from the y-axis of the best-fit ellipse to the selected region	
Average curvature	Reciprocal of the radius fit to every three consecutive pixels in the perimeter and averaged. (pixels^{-1})	

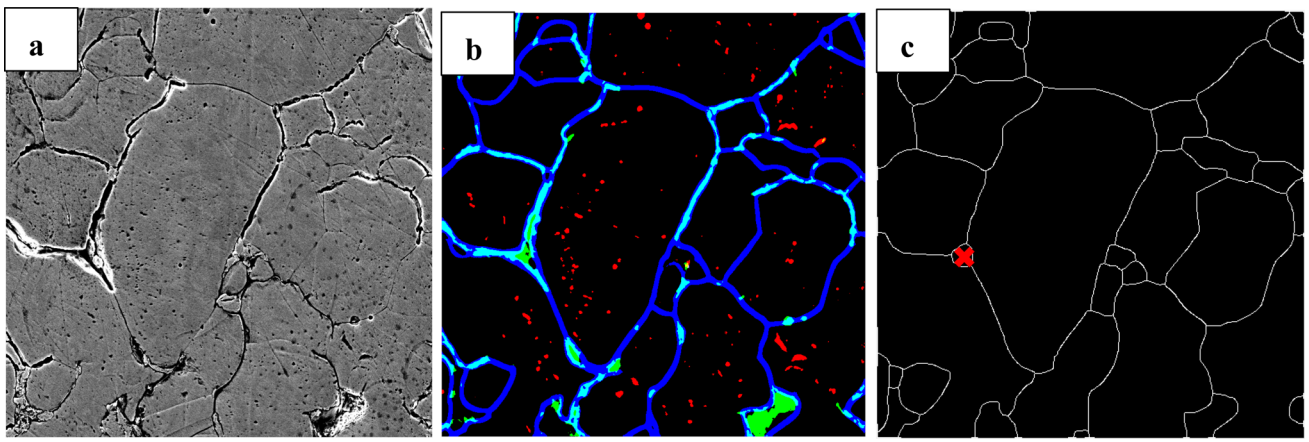


Fig. 24 **a** The input image, **b** masked to yield GB and pore mask, **c** grain regions that overlap with detected GB pores are removed

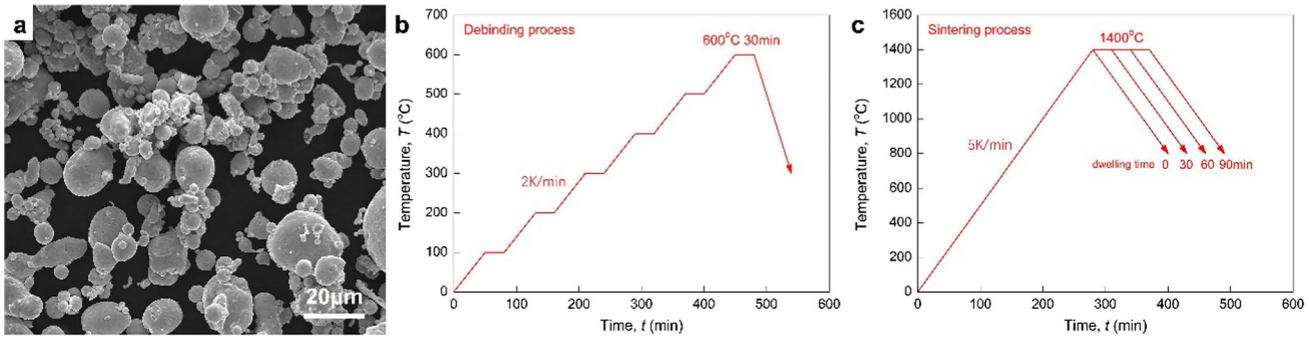


Fig. 25 The preparation routes for printed Ni samples: **a** SEM image of raw Ni powders; **b** temperature profile for debinding process and **c** temperature profile for sintering process

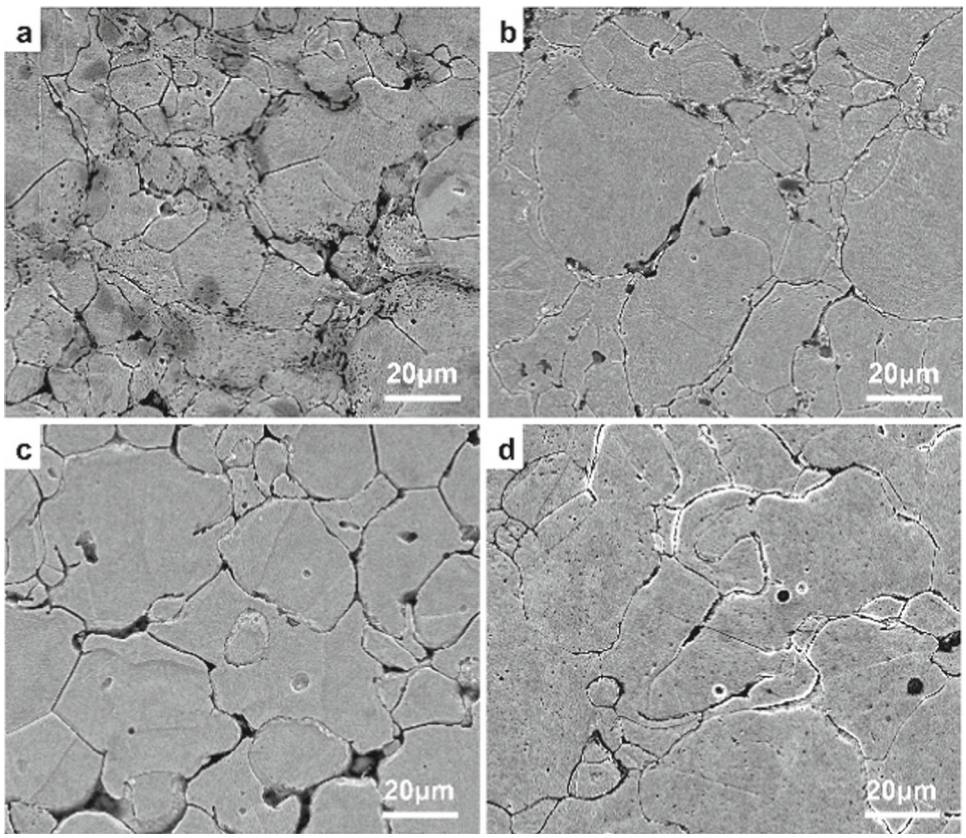


Fig. 26 Exemplary SEM images of printed Ni samples sintering at 1400oC with different dwell time: **a** 0min; **b** 30min; **c** 60min and **d** 90min

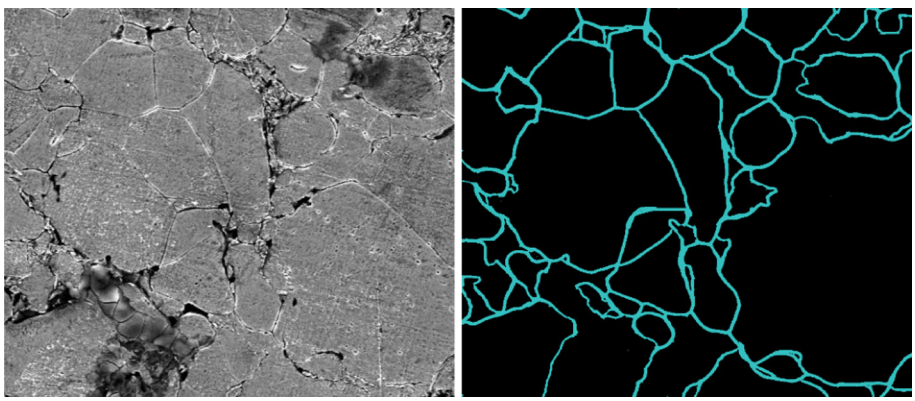


Fig. 27 Example of GB mask and image

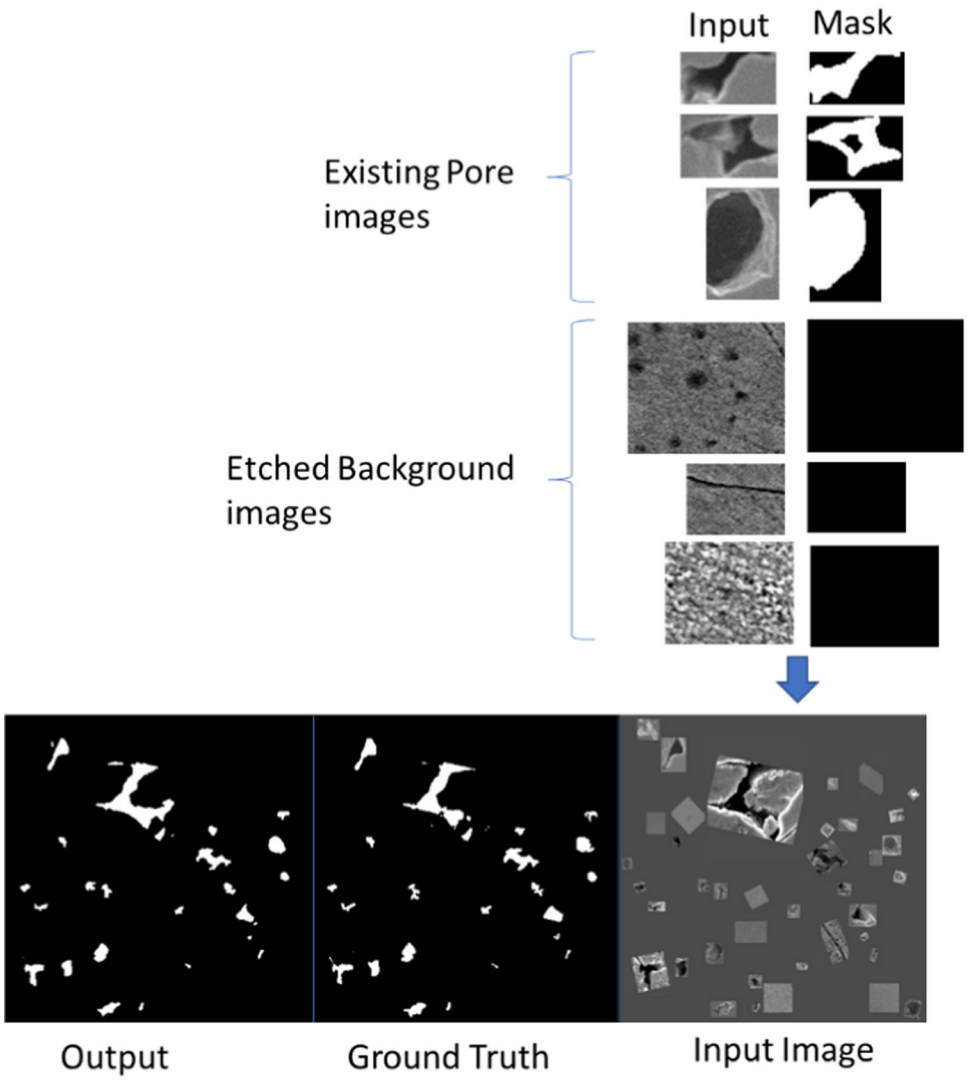


Fig. 28 Generation of training data for pore classifier, and trained pore classifier output

images, and therefore, pores cannot be manually labelled due to lack of human resources (i.e., trained experts). Instead, efforts in previous studies were leveraged to automate pore masking. The database previously developed in (Satterlee et al., 2022) included examples cropped images of pores and non-pores. The masking algorithm used in (Satterlee et al., 2023) was applied to the pore images generate a mask. The resulting database contained over 8000 examples of pores and non-pore images. However, as discussed in "Approach" Sect., there are many examples of etched images containing pore features in non-pore areas. Therefore, examples of these instances were manually cropped and included into the dataset with a blank mask. Finally, instances of the pore images, non-pore images, and the background images were randomly selected to generate a collage. The classifier is then trained on these generated images and masks to perform pore detection. An example of this process is presented in Fig. 28.

The collage was quickly generated in large batches by employing a second GPU to construct the images. This second GPU starts with a blank matrix and performs convolutions on the image with the kernel the same size as the training data. For areas where the convolution is zero, the area is empty and is a candidate location to insert the image. The candidate location is randomly selected, and image insertion continues until there are no spots remaining for a new image. In addition, inserted images were randomly rotated to prevent overfitting, however, resizing was not performed so that distinctions between pores and GBs based on size and aspect ratio could be maintained.

The trained classifier can now mask pores with high accuracy but performs poorly without localization of pores. To localize the pores in the etched image dataset, the general pore detection model generated in (Satterlee et al., 2023) was employed. The image is processed by the pore detection

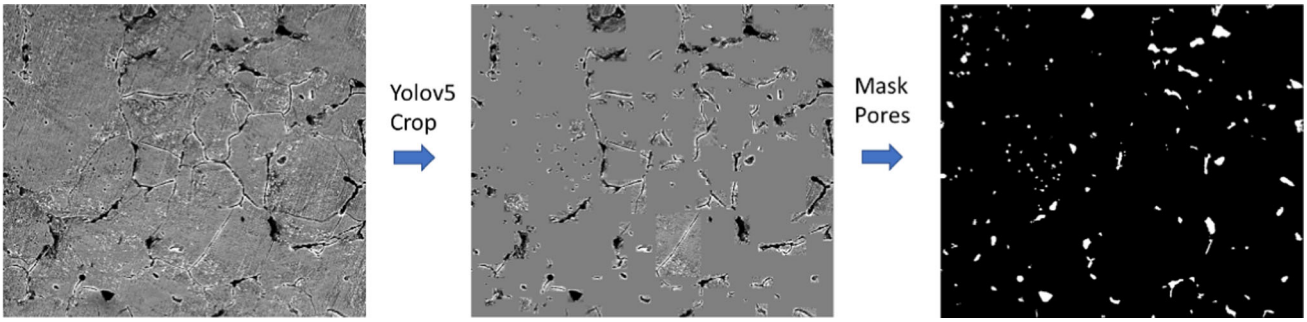


Fig. 29 Yolov5 pore localization, cropping of pores, and masking of pores

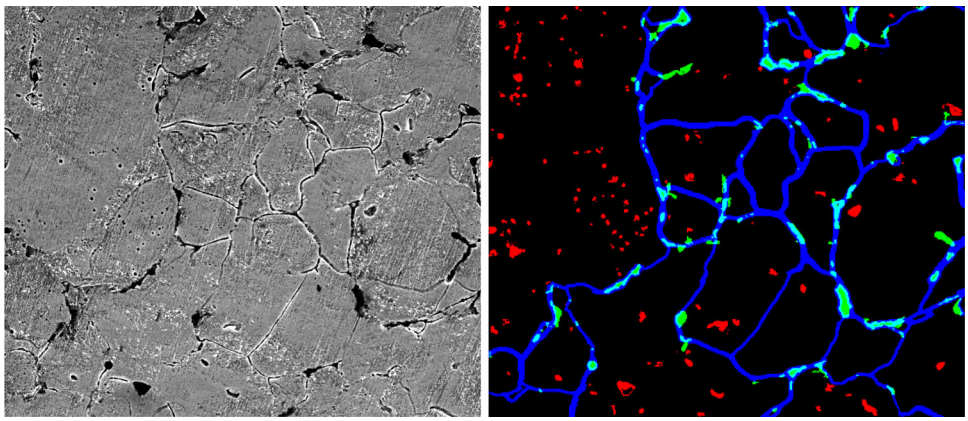


Fig. 30 Input image (left) and final mask (right) with internal pores (red), GB pores (green), and GB (blue), and grains (black)

model and the image is cropped removing all non-pore locations. The pore classifier is then run on the cropped image to obtain a ground-truth pore mask. An example of his process is shown in Fig. 29. Note that the yolov5 model was not trained on images with extensive etching and thus tends to classify GBs as pores, however, by including GBs in the masking model, these GBs are not present in the final mask. After pore masking was complete, images were manually edited to fix masking errors.

With the ground truth GB mask and pore mask data generated, they are then combined to create the final ground truth segmentation mask. Pores lying on GBs are given a separate classification from pores within the grains. An example of a final ground truth segmentation mask is shown in Fig. 30.

Training and testing data set

Initially 15% of the images were set aside for testing the machine learning models so that they would remain independent from the images generated by the diffusion network. The remaining images were used to train the diffusion network and the trained network was used to double the image set. The generated images were then manually vetted and those that appeared to be outliers were removed. In total, 81 images were selected for testing and 745 images were in the training set (of those 312 are generated images).

The clear GB images were found from publications and public manufacturing sites. In total, 45 clear GB images with a large variety were included in the testing dataset. Modifications to the images were performed as necessary, such as cropping out multiple images listed in a single figure. Image rescaling is automatically performed to match the training size.

Discussion of the results

After each network was trained with the prescribed training scheme, the threshold was then adjusted to maximize the accuracy on the validation set, then the resulting parameter was used to evaluate the test set for that network. The results of each network on the unclear testing images are listed in Table 3 and some examples of result images are provided in Fig. 31. The results suggest that the UNet with 64 initial layers performs best on the dataset, though the other networks do not trail behind by much. Note that the IOU is reported, but its meaning has little significance since the GB is ultimately skeletonized.

After investigating the data, it is found that 90% of the images yielded an accuracy above 80% as shown in Fig. 32. This means that a few outliers are weighting the results. All the classifiers share the same image outliers. The output of

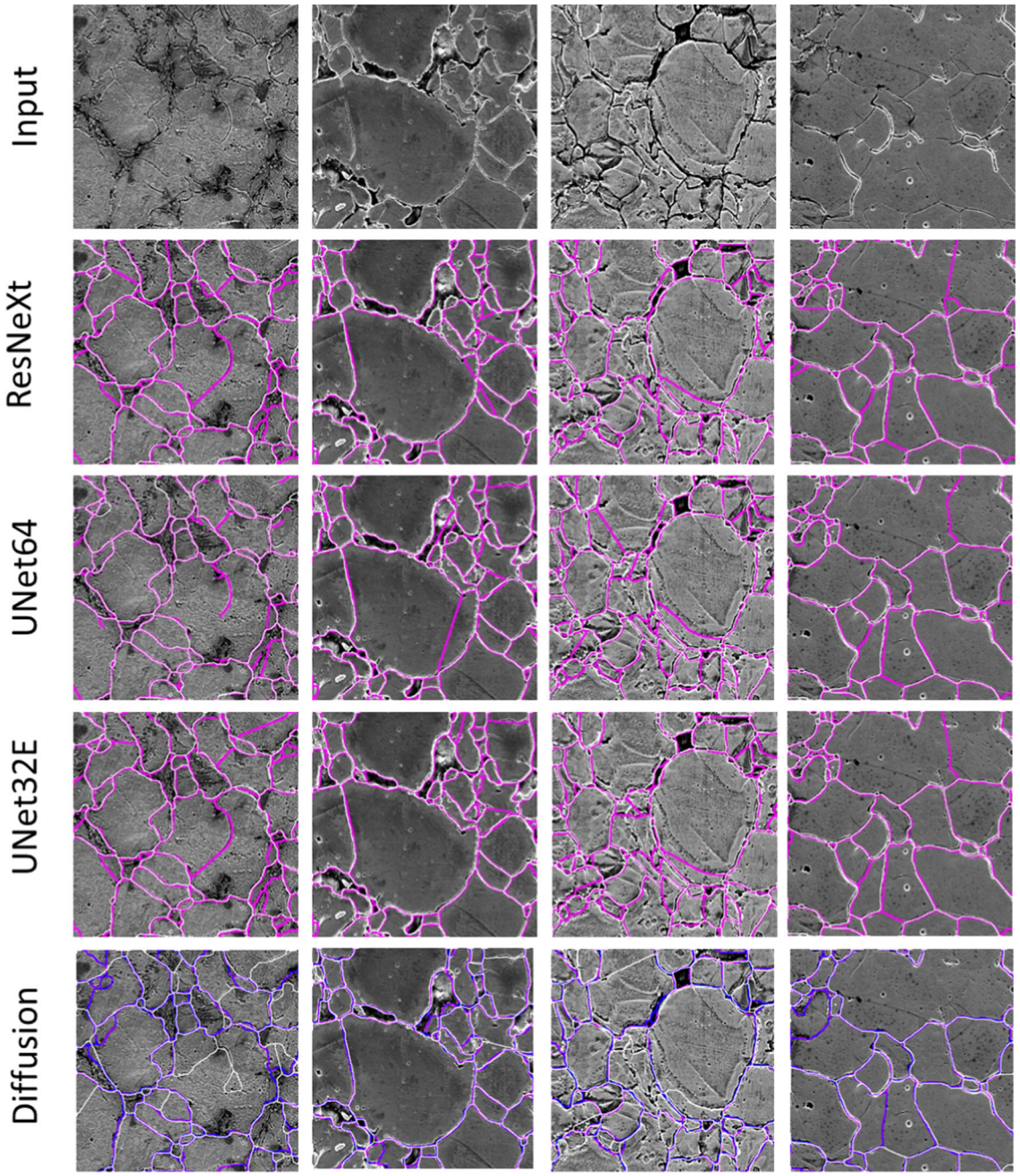


Fig. 31 Examples of predicted GBs with blue (predicted) and white (ground-truth) for each network

each outlier and the input image is displayed in Fig. 33. It is seen that the outliers contain ambiguous GBs which makes labelling them challenging. The models are then more likely to predict discontinuous GBs resulting in increased interpolation. Interestingly, the ResNeXt model appears to perform very well by visual inspection and may make better decisions than the ground truth mask on the images. Due to the labelling ambiguity of these images, the performance without the outliers is displayed in Table 4. Eliminating these four outliers brings the accuracy above 90% for the ResNeXt model.

The networks were then run on the test images found in literature with results displayed in Table 5. The ResNeXt model clearly outperforms the other models with grain size metrics above 92%. In this dataset few of the images contain pores, and the models tend to over predict pores due to the high

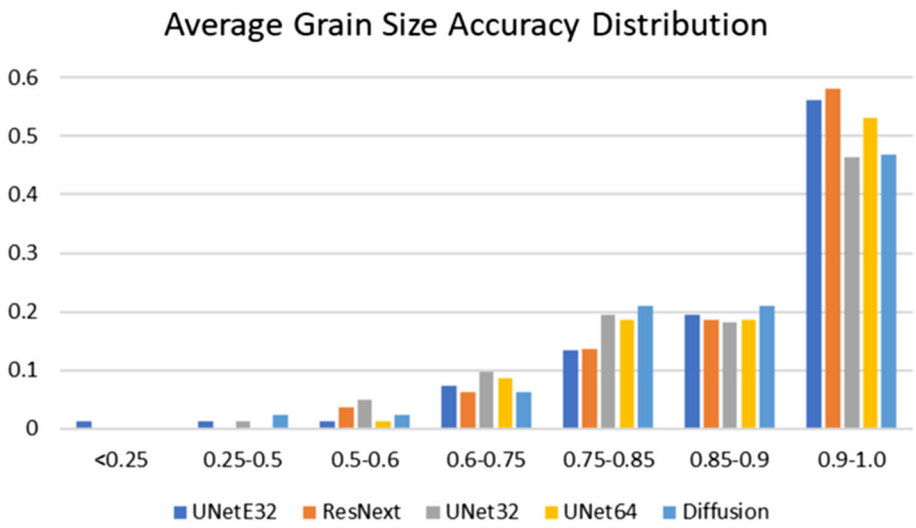


Fig. 32 Distribution of grain size accuracy for unclear image set

Table 3 Unclear image set accuracy results with the best results displayed in bold

Mean average	UNet32	UNet64	UNet32 ensemble	ResNeXt50	Denoising diffusion
Grain Size	0.860	**0.881**	0.877	0.878	0.872
Internal Pore Size	0.853	0.840	0.848	0.844	**0.873**
GB Pore Size	**0.852**	0.847	0.788	0.790	0.700
Grain Size Std Dev	0.846	**0.888**	0.880	0.886	0.869
Internal Pore Size Std Dev	0.741	**0.831**	0.829	0.827	0.791
GB Pore Size Std Dev	0.788	**0.805**	0.761	0.772	0.747
No. Grains	0.826	0.882	0.881	**0.889**	0.877
No. Internal Pores	0.726	0.800	0.867	**0.877**	0.861
No. GB Pores	0.713	0.827	0.802	**0.834**	0.546
Solidity	0.987	0.986	0.983	0.983	**0.988**
Aspect Ratio	0.938	**0.954**	0.953	0.953	0.932
Extent	0.979	**0.980**	0.978	0.977	0.978
Equivalent Diameter	0.927	0.928	**0.932**	0.926	0.923
Angle	0.913	**0.940**	0.934	0.934	0.901
Curvature	0.921	0.917	0.924	**0.927**	0.846
Contact Area	0.921	0.918	**0.921**	0.912	0.916
Length	0.917	0.921	**0.923**	0.914	0.916
IOU	0.654	0.674	**0.678**	0.660	0.620

porosity of the training images. Therefore, pores predictions were not reported on the found image set.

Next, we compare the performance of the networks with and without interpolation in Table 6. All networks tend to improve with the interpolation except the ensemble network. The results indicate that the residual networks and inverted network ensemble decreased GB discontinuity as intended. Further supporting the claim is the decrease in performance of the UNet64 and ResNeXt50 networks when interpolation is eliminated. This leads to the possibility of training the ResNeXt50 network along with a companion UNet32 to predict the residuals and perform interpolation. A potential improvement to the ensemble is the addition of a negative residual network to remove parts of the mask the are labelled incorrectly. To do this we simply swap the target and prediction in Eq. 3.

The diffusion network was also found to be unchanged with interpolation. However, the results of the test set are noticeably worse than the unclear image set. Upon examination, it was found the while the denoising diffusion network

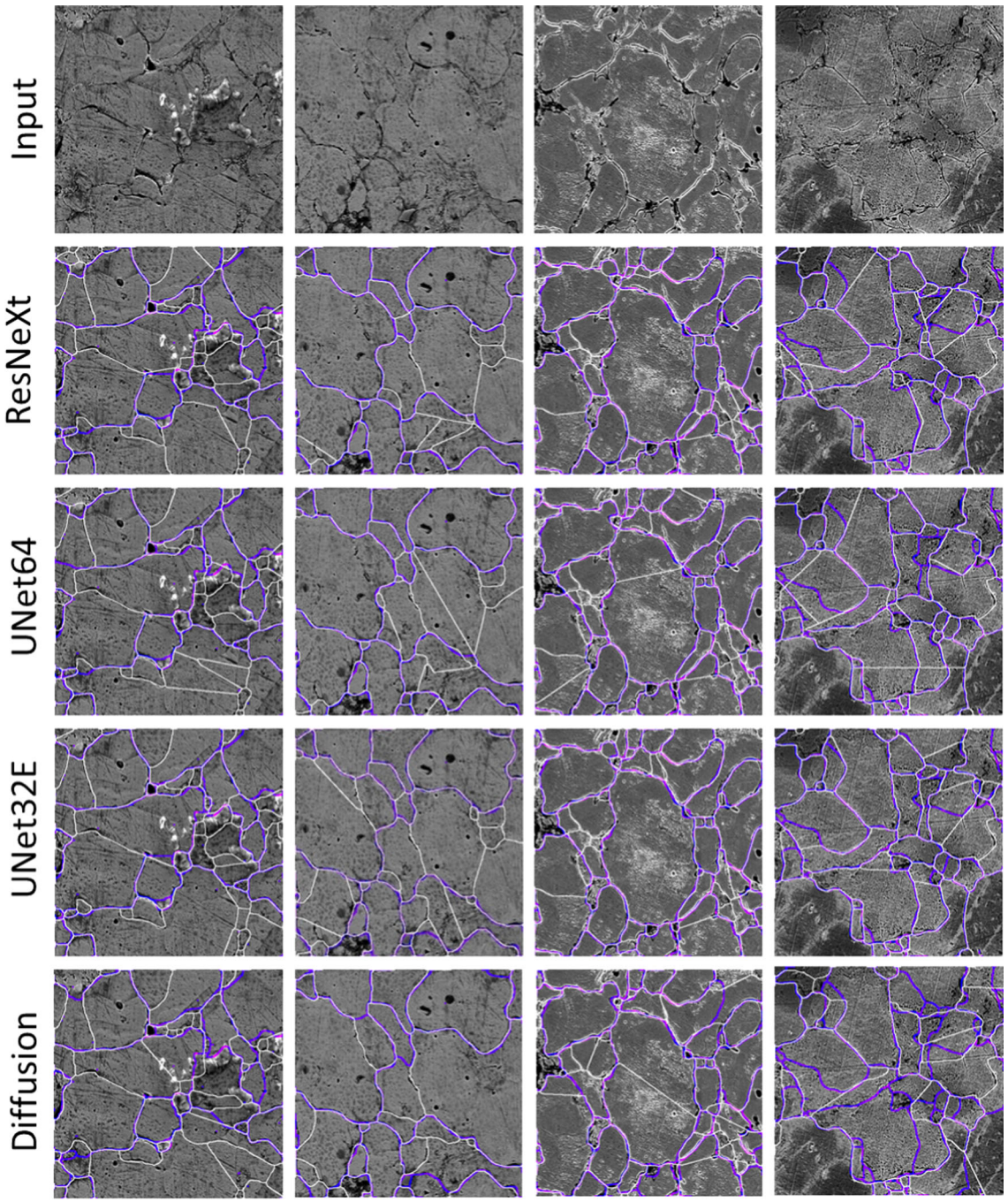


Fig. 33 Predicted GBs on outlier images with blue (ground truth) and white (predicted) for each network. Images are sorted by prediction accuracy with lowest at the top

exceeds in generating grain boundaries without discontinuities, it takes liberties in generating these boundaries when predicting on images outside the training domain as shown in Fig. 34. The number of denoising steps was varied (40, 50, 100, 150, and 250 steps) but false grain boundaries remained. It is noted that most of these clear grain boundary images are lower resolutions ($< 512 \times 512$) so it is unlikely that training at higher resolutions would improve results.

Discrepancies between the unclear image set and the clear image set are due to the complexity of the unclear images. Specifically, where GBs are located is a challenging problem for many of the images as shown in the outlier images from Fig. 33. Comparing these images to those found in literature from Figs. 35 and 36, the unclear images are far more difficult to label.

Table 4 Unclear image set accuracy results without outlier images. Best results are displayed in bold

Mean Average	UNet64	UNet32E	ResNeXt50	Diffusion
Grain Size	0.894	0.899	**0.902**	0.892
Grain Size Std Dev	0.902	0.895	**0.904**	0.880
No. Grains	0.895	0.900	**0.906**	0.892
Solidity	0.986	0.984	0.983	**0.987**
Aspect ratio	**0.956**	0.955	0.954	0.943
Extent	**0.980**	0.978	0.977	0.977
Equivalent diameter	0.937	**0.942**	0.938	0.912
Angle	**0.942**	0.934	0.934	0.915
Curvature	0.922	0.926	**0.930**	0.924
Contact Area	0.927	**0.933**	0.926	0.920
Length	0.931	**0.936**	0.927	0.924

Table 5 Test set accuracy results with best results displayed in bold. The ensemble network is denoted UNet32E

Mean average	UNet32	UNet64	UNet32E	ResNeXt50	Diffusion
Grain Size	0.889	0.904	0.908	**0.923**	0.820
Grain Size Std Dev	0.880	0.900	0.913	**0.922**	0.897
No. Grains	0.883	0.914	0.912	**0.922**	0.834
Solidity	0.994	0.994	0.994	**0.994**	0.992
Aspect ratio	0.941	0.946	0.937	0.945	**0.947**
Extent	0.970	0.972	0.977	0.977	**0.981**
Equivalent diameter	0.939	0.946	0.947	**0.954**	0.897
Angle	0.927	0.938	0.938	**0.943**	0.913
Curvature	0.893	0.881	0.878	**0.900**	0.847
Contact Area	0.930	0.937	0.939	**0.947**	0.887
Length	0.932	0.926	0.933	**0.941**	0.889

Table 6 Comparison of networks with and without interpolation with best results displayed in bold. Networks without interpolation are denoted wo/I

Mean average	UNet64		UNet32E		ResNeXt50		Diffusion	
	w/I	wo/I	w/I	wo/I	w/I	wo/I	w/I	wo/I
Grain Size	**0.904**	0.860	0.908	**0.925**	**0.923**	0.911	0.820	**0.822**
Grain Size Std Dev	**0.900**	0.751	**0.913**	0.891	**0.922**	0.854	**0.897**	**0.897**
No. Grains	**0.914**	0.829	**0.912**	0.906	**0.922**	0.881	0.834	**0.837**
Solidity	**0.994**	0.989	**0.994**	0.991	**0.994**	0.989	**0.992**	**0.992**
Aspect ratio	0.946	**0.949**	0.937	**0.956**	0.945	**0.959**	0.947	**0.948**
Extent	**0.972**	0.942	**0.977**	0.947	**0.977**	0.941	**0.981**	**0.981**
Equivalent diameter	0.946	**0.948**	0.947	**0.957**	0.954	**0.958**	0.897	**0.900**
Angle	**0.938**	0.932	0.938	**0.941**	0.943	**0.947**	**0.913**	**0.913**
Curvature	0.881	**0.907**	0.878	**0.895**	**0.900**	0.889	0.847	**0.854**
Contact Area	**0.937**	0.893	0.939	**0.943**	**0.947**	0.932	0.887	**0.890**
Length	**0.926**	0.916	0.933	**0.948**	0.941	**0.950**	0.889	**0.892**

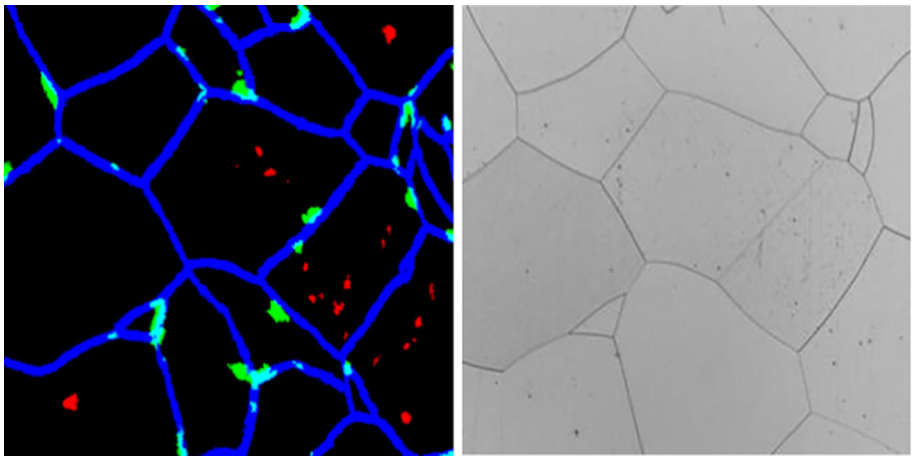


Fig. 34 Denoising diffusion mask prediction at 50 steps (left) and input image (right). Grain boundaries are continuous, but excessive grain boundaries are produced

Another observation is that the curvature tends to be more accurate in the unclear data. This is likely because the clear images are taken from literature, so they are in general much smaller than the training images. The masks are resized by the nearest neighbours to ensure that intensity values do not change. The nearest neighbour resizing will not maintain the original contour due to pixels being blocked together. Blocking does not occur when in the prediction mask since resizing is not necessary and thus the curvature prediction will vary.

To demonstrate the diversity of the testing set and the resilience of the methodology, testing images from each publication are displayed with the resulting mask in Figs. 35 and 36. Many publications contained a college of images that were cropped and processed independently. Since these images are generally similar to one another, only one was selected for display. The results show some errors. False GBs are marked as a result from unnecessary interpolation and true GBs are missed in some instances.

Conclusion

This study aims to develop a generalized grain boundary detection method for sintered 3D-printed metal parts which do not show clear grain boundaries with traditional etching. Because the traditional image processing algorithms and ML methods have limitations on those unclear GB images, the study developed general GB detection methods using deep learning, augmentation, and GB interpolation. The results show that the ensemble of UNets outperformed the other networks and predict all grain metrics with an accuracy around 90% for unclear GB images and 92% for clear GB images.

The results demonstrate that grain characterization can be automated quickly and accurately. Furthermore, it can be performed on unclear grain boundaries without developing optimized etching techniques for new materials. This reduces the overhead of prototyping with powder-based 3D-printed and sintered materials by reducing the requirement for refined etching.

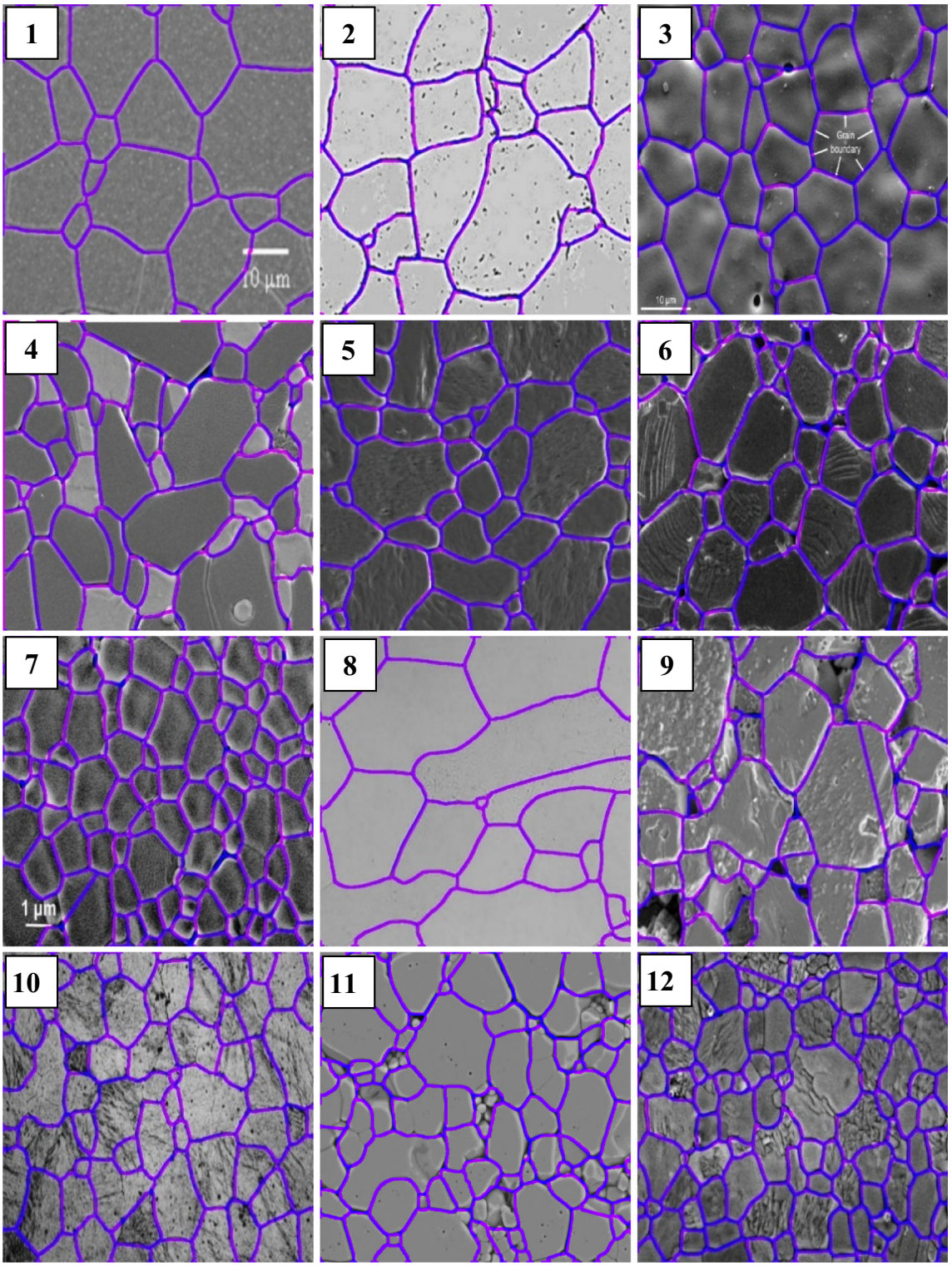


Fig. 35 Clear GB image set with GBs labelled in blue. (1) SiO2 doped YAG (Kochawattana et al., 2008), (2) Austenite (Yue et al., 2010), (3) Dense Polycrystalline Ceria (Bowman, 2016), (4) Zirconia-Toughened Alumina (Nanolytical, 2023), (5) ZnO (Dorraj et al., 2014), (6) Translucent barium titanate (Shimooka et al., 1998), (7) α-alumina (Voytovych et al., 2002), (8) Fe-3%Si steel (Fenghui et al., 2019), (9) Magnetite (Materials Chemistry, 2023), (10) AZ31 alloy (Du et al., 2008), (11) Tuite synthesized from chlorapatite (X. Xie et al., 2013), 12) fully yttri-astabilized cubic zirconia (Vagkopoulou et al., 2009)

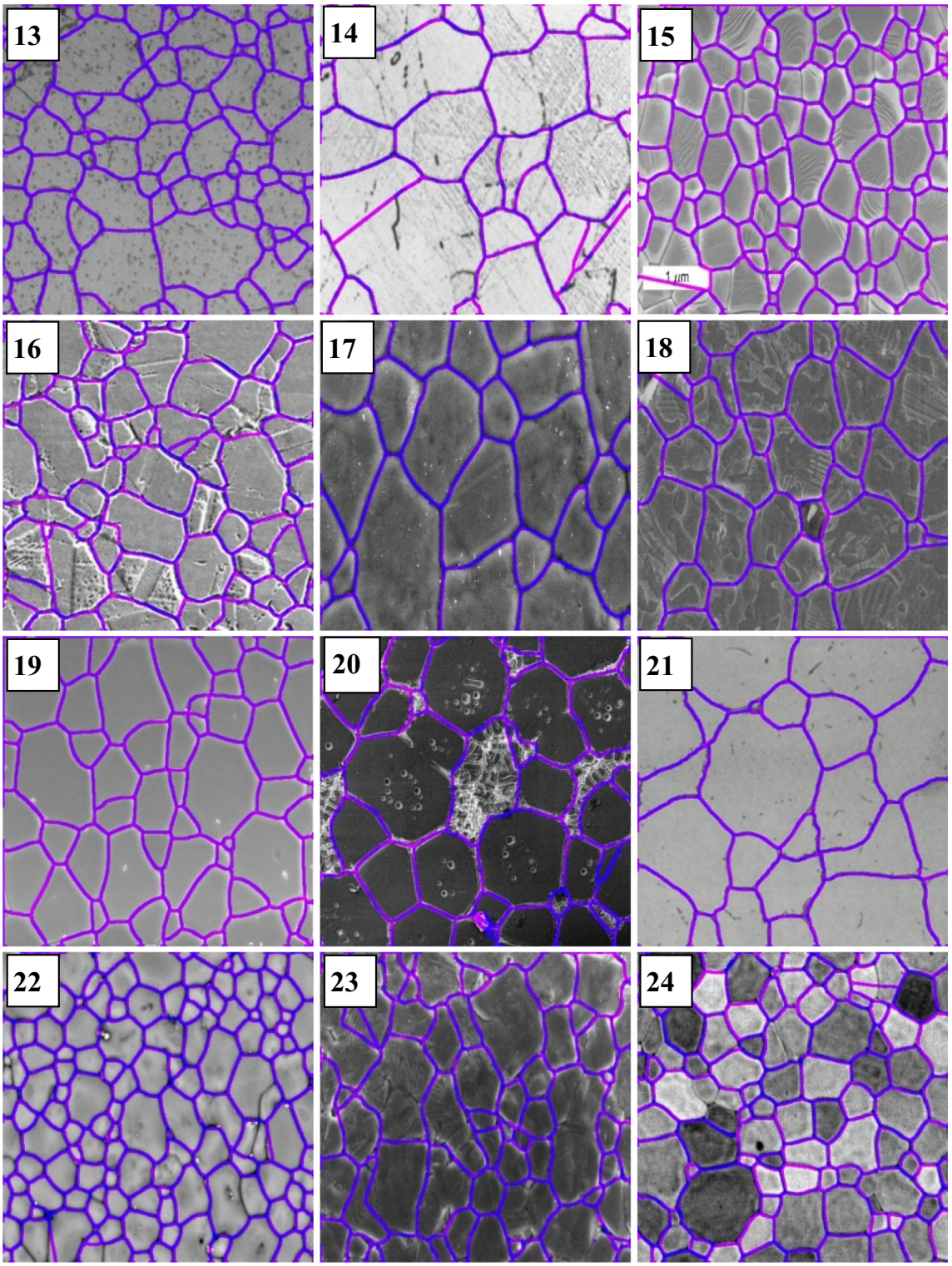


Fig. 36 Clear GB image set with GBs labelled in blue. (13) Cu-Be alloy (Hung et al., 1999), (14) 316 stainless steel (Shaikh et al., 2006), (15) Polycrystalline alumina (Harris et al., 2017), (16) 304 stainless steel (Garfias-Garcia et al., 2010), (17) CdTe film on Mo-glass (Valdna et al., 2010), (18) PZT polycrystals (Kozinov & Kuna, 2018), (19) Lead magnesium niobato (Londoño et al., 2012), (20) 7075 Al alloy (Binesh & Aghaie-Khafri, 2016), (21) Alloy 600 (H. P. Kim et al., 2017), (22) BFO-BTO-Mn ceramic (Chen et al., 2015), (23) Ceramic BETZ (Si-Ahmed et al., 2018), (24) Sintered ThO2 (Ray et al., 2012)

Future work will focus on grain boundary area evolution kinetics by developing a process to quickly assess the grain boundary area evolution given a series of microstructure images. However, it is expected that further testing and finetuning of the model as well as data collection requirements will be necessary since error accumulation is expected when combining predictions to perform kinetic analysis. We expect the general GB detection method developed in this study will contribute to the characterization of sintered 3D printed metal parts.

Funding This work is supported by the National Science Foundation (Grant No. 2119832).

References

Armstrong, M., Mehrabi, H., & Naveed, N. (2022). An overview of modern metal additive manufacturing technology. *Journal of Manufacturing Processes*. https://doi.org/10.1016/j.jmapro.2022.10.060

Binesh, B., & Aghaie-Khafri, M. (2016). Phase evolution and mechanical behavior of the semi-solid SIMA processed 7075 aluminum alloy. *Metals, 6*(3), 42. https://doi.org/10.3390/met6030042

Bordas, A., Zhang, J., & Nino, J. C. (2022). Application of deep learning workflow for autonomous grain size analysis. *Molecules, 27*(15), 4826. https://doi.org/10.3390/molecules27154826

Bowman, W. J. (2016). *Correlating nanoscale grain boundary composition with electrical conductivity in ceria*. Arizona State University.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.1986.4767851

Catania, R., Senthilnathan, A., Sions, J., Snyder, K., Al-Ghaib, H., Zimmerman, B., & Acar, P. (2022). New methodologies for grain boundary detection in EBSD data of microstructures. *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*. https://doi.org/10.2514/6.2022-1424

Chen, Y., Mei, K., Wong, C. M., Lin, D., Chan, H. L. W., & Dai, J. (2015). Ultrasonic transducer fabricated using lead-free BFO-BTO+Mn piezoelectric 1–3 composite. *Actuators, 4*(2), 127–134. https://doi.org/10.3390/act4020127

Community, O. (2010, Oct). The OpenCV reference manual.

Decost, B. L., & Holm, E. A. (2015). A computer vision approach for automated analysis and classification of microstructural image data. *Computational Materials Science, 110*, 126–133. https://doi.org/10.1016/j.commatsci.2015.08.011

Dengiz, O., Smith, A. E., & Nettleship, I. (2005). Grain boundary detection in microstructure images using computational intelligence. *Computers in Industry, 56*(8–9), 854–866. https://doi.org/10.1016/j.compind.2005.05.012

Dorraj, M., Zakaria, A., Abdollahi, Y., Hashim, M., & Moosavi, S. (2014). Optimization of Bi_2O_3, TiO_2, and Sb_2O_3 doped ZnO-based low-voltage varistor ceramic to maximize nonlinear electrical properties. *The Scientific World Journal*. https://doi.org/10.1155/2014/741034

Du, J., Yang, J., Kuwabara, M., Li, W., & Peng, J. (2008). Effects of carbon and/or alkaline earth elements on grain refinement and tensile strength of AZ31 alloy. *Materials Transactions, 49*(10), 2303–2309. https://doi.org/10.2320/matertrans.MRA2008146

Fenghui, A., Bo, L., Deqin, Z., Jinlong, L., & Yuhui, S. (2019). Recrystallization kinetics of Fe-3%Si after deformation at high strain rate and high temperature. *Materials Research, 22*(suppl 2), e20180746. https://doi.org/10.1590/1980-5373-mr-2018-0746

Friel, J. J., Prestridge, E. B., & Glazer, F. (1990). Advances in video technology for microstructural control. *Grain boundary reconstruction for grain sizing.* American Society for Testing and Materials.

Garfias-Garcia, E., Colin-Paniagua, F. A., Herrera-Hernández, H., Juarez-Garcia, J. M., Palomar-Pardavé, M. E., & Romero-Romo, M. R. (2010). Electrochemical and microscopy study of localized corrosion on a sensitized stainless steel AISI 304. *ECS Transactions, 29*(1), 93–102. https://doi.org/10.1149/1.3532307

Gupta, S., Sarkar, J., Banerjee, A., Bandyopadhyay, N. R., & Ganguly, S. (2019). Grain boundary detection and phase segmentation of SEM Ferrite–Pearlite microstructure using SLIC and skeletonization. *Journal of the Institution of Engineers (India): Series D*, *100*(2), 203–210. https://doi.org/10.1007/s40033-019-00194-1

Harris, D. C., Johnson, L. F., Cambrea, L., Baldwin, L., Baronowski, M., Zelmon, D. E., Poston, W. B., Kunkel, J. D., Parish, M., Pascucci, M. R., Gannon, J. J., & Wen, T.-C. (2017). Refractive index of infrared-transparent polycrystalline alumina. *Optical Engineering*. https://doi.org/10.1117/1.oe.56.7.077103

Ho, J., Jain, A., & Abbeel, P. (2020, Dec). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*.

Hung, N. P., Zhong, Z. W., Lee, K. K., & Chai, C. F. (1999). Precision grinding and facing of copper-beryllium alloys. *Precision Engineering*. https://doi.org/10.1016/S0141-6359(99)00024-0

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., & Aila, T. (2021). *Alias-free generative adversarial networks*. arXiv. https://doi.org/10.48550/ARXIV.2106.12423

Khalifa, N. E., Loey, M., & Mirjalili, S. (2022). A comprehensive survey of recent trends in deep learning for digital images augmentation. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-021-10066-4

Kim, H. P., Choi, M. J., Kim, S. W., Kim, D. J., Lim, Y. S., & Hwang, S. S. (2017). Effects of grain boundary morphologies on stress corrosion cracking of alloy 600. *Archives of Metallurgy and Materials*. https://doi.org/10.1515/amm-2017-0219

Kim, J. S., & Cho, H. S. (1994). A fuzzy logic and neural network approach to boundary detection for noisy imagery. *Fuzzy Sets and Systems, 65*, 141–159.

Kochawattana, S., Stevenson, A., Lee, S. H., Ramirez, M., Gopalan, V., Dumm, J., Castillo, V. K., Quarles, G. J., & Messing, G. L. (2008). Sintering and grain growth in SiO_2 doped Nd:YAG. *Journal of the European Ceramic Society, 28*(7), 1527–1534. https://doi.org/10.1016/j.jeurceramsoc.2007.12.006

Kozinov, S., & Kuna, M. (2018). Simulation of fatigue damage in ferroelectric polycrystals under mechanical/electrical loading. *Journal of the Mechanics and Physics of Solids*. https://doi.org/10.1016/j.jmps.2018.03.013

Lai, Z., Duan, Y., Dai, J., Li, Z., Fu, Y., Li, H., Qiao, Y., Wang, W. (2023). Denoising diffusion semantic segmentation with mask prior modeling. *ArXiv*.

Li, M., Chen, D., & Liu, S. (2020). Grain boundary detection based on multi-level loss from feature and adversarial learning. *IEEE Access, 8*, 135640–135651. https://doi.org/10.1109/ACCESS.2020.3011703

Londoño, F. A., Eiras, J., Milton, F. P., & Garcia, D. (2012). Preparation and microstructural, structural, optical and electro-optical properties of La doped Pmn-Pt transparent ceramics. *Optics and Photonics Journal, 02*(03), 157–162. https://doi.org/10.4236/opj.2012.23023

Materials chemistry. (2023). Retrieved from https://web.mit.edu/allanore/www/websiteallanoregroup/science.html

Modi, S., Lin, Y., Cheng, L., Yang, G., Liu, L., & Zhang, W. J. (2011). A socially inspired framework for human state inference using expert

opinion integration. *IEEE/ASME Transactions on Mechatronics, 16*(5), 874–878. https://doi.org/10.1109/TMECH.2011.2161094

Mostafaei, A., Elliott, A. M., Barnes, J. E., Li, F., Tan, W., Cramer, C. L., Nandwana, P., & Chmielus, M. (2021). Binder jet 3D printing—process parameters, materials, properties, modeling, and challenges. *Progress in Materials Science, 119*, 100707. https://doi.org/10.1016/j.pmatsci.2020.100707

Nanolytical. (2023). Materials testing. https://nanolytical.com.au/materials-testing/

Olevsky, E. A., & Dudina, D. V. (2018). Field-assisted sintering: Science and applications. *Field-Assisted Sintering Science and Applications*. https://doi.org/10.1007/978-3-319-76032-2

Ray, A., Banerjee, J., Kutty, T., Kumar, A., & Banerjee, S. (2012). Construction of master Sintering Curve of ThO_2 pellets using optimization technique. *Science of Sintering*. https://doi.org/10.2298/SOS1202147R

Rohrer, G. S. (2011). Grain boundary energy anisotropy: A review. *Journal of Materials Science, 46*(18), 5881–5895. https://doi.org/10.1007/s10853-011-5677-3

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), 9351*. https://doi.org/10.1007/978-3-319-24574-4_28

Satterlee, N., Torresani, E., Olevsky, E., & Kang, J. S. (2022). Comparison of machine learning methods for automatic classification of porosities in powder-based additive manufactured metal parts. *The International Journal of Advanced Manufacturing Technology, 120*(9), 6761–6776. https://doi.org/10.1007/s00170-022-09141-z

Satterlee, N., Torresani, E., Olevsky, E., & Kang, J. S. (2023). Automatic detection and characterization of porosities in cross-section images of metal parts produced by binder jetting using machine learning and image augmentation. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02100-9

Shaikh, H., Sivaibharasi, N., Sasi, B., Anita, T., Amirthalingam, R., Rao, B. P. C., Jayakumar, T., Khatak, H. S., & Raj, B. (2006). Use of eddy current testing method in detection and evaluation of sensitisation and intergranular corrosion in austenitic stainless steels. *Corrosion Science, 48*(6), 1462–1482. https://doi.org/10.1016/j.corsci.2005.05.017

Shimooka, H., Yamada, K. I., Takahashi, S., & Kuwabara, M. (1998). Preparation of transparent, partially-crystallized BaTiO3 monolithic xerogels by SOL-GEL PROCESSING. *Journal of Sol-Gel Science and Technology, 13*(1–3), 873–876. https://doi.org/10.1023/a:1008662904874

Si-Ahmed, F., Taïbi, K., Bidault, O., & Millot, N. (2018). Dielectric behavior of a lead-free electroceramics Ba1−xEr2x/3(Ti1−yZry)O3. *Journal of Materials Science: Materials in Electronics, 29*(12), 10154–10163. https://doi.org/10.1007/s10854-018-9061-9

Tony, A., Badea, I., Yang, C., Liu, Y., Wells, G., Wang, K., Yin, R., Zhang, H., & Zhang, W. (2023). The additive manufacturing approach to polydimethylsiloxane (PDMS) microfluidic devices: Review and future directions. *Polymers, 15*(8), 1926. https://doi.org/10.3390/polym15081926

Vagkopoulou, T., Koutayas, S. O., Koidis, P., & Strub, J. R. (2009). Zirconia in dentistry: Part 1. Discovering the nature of an upcoming bioceramic. *The European Journal of Esthetic Dentistry: Official Journal of the European Academy of Esthetic Dentistry,* 4(2).

Valdna, V., Grossberg, M., Hiie, J., Kallavus, U., Mikli, V., Traksmaa, R., & Viljus, M. (2010). Preparation and properties of CdTe films on Mo/Glass substrates. *Materials Research Society Symposium Proceedings*. https://doi.org/10.1557/proc-1165-m08-24

Voytovych, R., MacLaren, I., Gülgün, M. A., Cannon, R. M., & Rühle, M. (2002). The effect of yttrium on densification and grain growth in α-alumina. *Acta Materialia, 50*(13), 3453–3463. https://doi.org/10.1016/S1359-6454(02)00159-3

Wei, Y., Peng, Z., Kühbach, M., Breen, A., Legros, M., Larranaga, M., Mompiou, F., & Gault, B. (2019). 3D nanostructural characterisation of grain boundaries in atom probe data utilising machine learning methods. *PLoS ONE, 14*(11), e0225041. https://doi.org/10.1371/journal.pone.0225041

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings—30th IEEE conference on computer vision and pattern recognition, CVPR 2017*. https://doi.org/10.1109/CVPR.2017.634

Xie, X., Zhai, S., Chen, M., & Yang, H. (2013). Tuite, γ-Ca3(PO4)2, formed by chlorapatite decomposition in a shock vein of the Suizhou L6 chondrite. *Meteoritics and Planetary Science, 48*(8), 1515–1523. https://doi.org/10.1111/maps.12143

Yue, C., Zhang, L., Liao, S., & Gao, H. (2010). Kinetic analysis of the austenite grain growth in GCr15 steel. *Journal of Materials Engineering and Performance, 19*(1), 112–115. https://doi.org/10.1007/s11665-009-9413-y

Zhang, T. Y., & Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM, 27*(3), 236–239. https://doi.org/10.1145/357994.358023

Zhang, W. J., Yang, G., Lin, Y., Ji, C., & Gupta, M. M. (2018). On definition of deep learning. *World Automation Congress Proceedings*. https://doi.org/10.23919/WAC.2018.8430387