Deep Learning-based Semantic Image Segmentation Techniques for Corrosive Particles of Aluminum Alloy AA 7075

by

Daniel Barboza

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved September 2020 by the
Graduate Supervisory Committee:

Pavan Turaga, Chair
Nikhilesh Chawla
Suren Jayasuriya

ARIZONA STATE UNIVERSITY

December 2020

ABSTRACT

Semantic image segmentation has been a key topic in applications involving image processing and computer vision. Owing to the success and continuous research in the field of deep learning, there have been plenty of deep learning-based segmentation architectures that have been designed for various tasks. In this thesis, deep-learning architectures for a specific application in material science; namely the segmentation process for the non-destructive study of the microstructure of Aluminum Alloy AA 7075 have been developed. This process requires the use of various imaging tools and methodologies to obtain the ground-truth information. The image dataset obtained using Transmission X-ray microscopy (TXM) consists of raw 2D image specimens captured from the projections at every beam scan. The segmented 2D ground-truth images are obtained by applying reconstruction and filtering algorithms before using a scientific visualization tool for segmentation. These images represent the corrosive behavior caused by the precipitates and inclusions particles on the Aluminum AA 7075 alloy. The study of the tools that work best for X-ray microscopy-based imaging is still in its early stages. In this thesis, the underlying concepts behind Convolutional Neural Networks (CNNs) and state-of-the-art Semantic Segmentation architectures have been discussed in detail. The data generation and pre-processing process applied to the AA 7075 Data have also been described, along with the experimentation methodologies performed on the baseline and four other state-of-the-art Segmentation architectures that predict the segmented boundaries from the raw 2D images. A performance analysis based on various factors to decide the best techniques and tools to apply Semantic image segmentation for X-ray microscopy-based imaging was also conducted.

ACKNOWLEDGMENTS

I would like to especially thank Professor Pavan Turaga, for giving me a wonderful opportunity to work on my Master's thesis under his guidance. He has been completely supportive throughout his valuable suggestions and insights. I would also like to thank him for his patience with me and for putting up with my slow progress over time.

I would also like to express my special thanks to Rajhans Singh for helping me through the research with his valuable inputs and assisting me with technical difficulties and approaches related to the topic. Also, I wish to acknowledge Kowshik Thopalli for assisting me with my queries. It was great being a part of the Geometric Media Lab at Arizona State University (ASU) and contributing to research work in the time frame possible.

I am grateful to Professor Suren Jayasuriya and Professor Nikhilesh Chawla for being willing to serve as panelists for the Thesis defense committee. I would also like to thank Dr. Sridhar Niverty and the Center for 4D Material Sciences, ASU directed by Professor Nikhilesh for granting me access to the dataset for research.

I also wish to thank my graduate advisor, Lynn Pratte, from the School of Electrical, Computer and Energy Engineering at ASU, for assisting me on various occasions in my Master's program and for her kind advice.

Finally, I wish to thank my parents, sister, best friends, and roommates for their constant support and encouragement throughout my Master's program at ASU.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

X-ray microscopy is the process of producing magnified images of specimen objects using electromagnetic radiations. The Aluminum Alloy AA 7075 is a complex heterogeneous alloy containing particles such as metallic inclusions and corrosion that contribute to the corrosion of the alloy. Using Transmission X-ray Microscopy (TXM), it is possible to study the corrosion behavior of the alloy caused by the presence of other particles. The study also involves the use of image segmentation techniques with the help of scientific evaluation tools to quantify and visualize the distribution of the precipitates and inclusions contributing to the damage caused by corrosion. With the advances in Semantic segmentation approaches using deep-learning, it has been possible to effectively segment various classes related to images from the medical domain.

Semantic image segmentation involves the processing of an image to infer semantic labels for each pixel depending upon the class it belongs to. With recent progress in Deep learning and the use of GPU's with high computational capability, there has been a substantial amount of work aimed at developing semantic segmentation approaches using Deep Convolutional networks (Zhao and Xie (2013)). It has been possible to train deep-learning models with large datasets to obtain state-of-the-art performance for Computer Vision tasks like Object Detection, Image segmentation, Instance segmentation, etc (Minaee *et al.* (2020)).

Based on the advances in deep-learning and potential for its application to X-ray microscopy images, I have approached this task of performing Image segmentation on images from the domain of Material science using deep-learning. The chosen approaches have been able to achieve significant performance on image data from different domains (Liu *et al.* (2019)). This problem requires the use of advanced techniques to obtain accurately segmented particle boundaries which can be achieved with the help of deep-learning models. The research I have conducted focuses on the usage of state-of-the-art deep learning architectures for the task of Semantic segmentation to achieve a modest improvement in performance over our chosen baseline methodology. It discusses and comparatively analyzes the different methods chosen for Semantic segmentation based on the performance and training metrics.

1.2 Objective

The primary objective of this thesis is to employ deep-learning techniques for the task of semantic image segmentation on X-ray microscopy images. For an optimal segmentation output, we also need to identify the ideal set of hyperparameters, methods, backbones, and architectures ideal for training and prediction upon the dataset. The time and memory consumption aspect of the chosen methods is also an important objective of this thesis to achieve the best possible segmentation results using an optimal amount of time and resources.

1.3 Thesis Outline

This thesis is divided into five parts, Chapter 1 introduces the problem, the motivation to solve the problem and describes the research method. Chapter 2 discusses the literature survey conducted for the task of Semantic image segmentation to identify potential deep-learning approaches apt for the task at hand. We discuss the work most relevant to our problem starting from traditional methods to state-of-the-art Convolutional Neural Network (CNN) architectures. Chapter 3 covers the background theory and concepts related to CNN’s in detail and discusses four of the commonly used Deep Neural architectures for Semantic segmentation in detail. Chapter 4 introduces the dataset of Aluminum AA 7075 images used for the research and other techniques employed to aid in training them using various approaches. It also includes the experiments conducted and analysis obtained from the results in comparison with the baseline model and each of the chosen architectures. Chapter 5 concludes the research with a summary of our contributions to the task of applying Semantic image segmentation to X-ray microscopy images and also discusses some future research directions.

Chapter 2

LITERATURE REVIEW

Semantic image segmentation deals with the assignment of class labels for every pixel of an image based on the class it belongs to (Ikeuchi (2014)). It has multiple applications in the fields of medical imaging and autonomous vehicles. Segmentation has been widely used to classify biomedical images to segment neuron structures (Ronneberger *et al.* (2015)), detection of brain tumor (Moon *et al.* (2002)), for the purpose of colon crypt segmentation (Cohen *et al.* (2015)), etc. The segmentation of different objects is significant in the autonomous domain and such methods, described by Maldonado-Bascón *et al.* (2007) and Gupta and Sortrakul (1998), have been used time and again before the use of neural network architectures.

Traditional methods like k-means clustering, thresholding, histogram-based methods have been employed for Semantic segmentation for various applications. However, over the course of years, deep-learning has pushed the boundaries to develop various architectures capable of enhancing the segmentation accuracy significantly. With the use of Deep convolutional neural networks (DCNN), it has been possible to achieve state-of-the-art performance for various visual tasks.

Krizhevsky *et al.* (2012) pioneered the current advancements in the field of deep-learning by performing supervised training of a large network on the ImageNet dataset with 1 million training images. Different deep architectures modified for training in different domains have been introduced since the advancement of deep learning-based segmentation methods. A network with a sliding window setup to predict pixel labels was suggested by Ciresan *et al.* (2012) which was slow in processing and less accurate. Various other implementation involved the use of features from different layers of the architecture as discussed in Seyedhosseini *et al.* (2013). Ronneberger *et al.* (2015) introduced an encoder-decoder type of architecture for biomedical image segmentation to improve localization accuracy.

Relevant work did by Chen *et al.* (2017), to add fully-connected random fields to CNNs led to a significant upgrade in the segmentation performance. Various other approaches by Zhao *et al.* (2017) involving the use of a pyramid architecture to concatenate various feature maps also proved well. The DeepLab v1 (Chen *et al.* (2017)) and DeepLab v2 (Chen *et al.* (2014)) paved the way for DeepLab v3+ (Chen *et al.* (2018)) which incorporates advanced elements from the previous two implementations.

As discussed above, there are a variety of potential methods to be implemented for the task of semantic segmentation. In this case, my approach would be to apply deep learning-based Semantic segmentation to segment inclusions and precipitates in images of Aluminum Alloy AA 7075. These images generated using the process of X-ray microscopy seem similar to images used for segmentation in the biomedical domain. There has been some research pertaining to segmentation for X-ray microscopy images as discussed by Kaira *et al.* (2018), Wang (2008), Ma *et al.* (2018), and Chen *et al.* (2020).

In this section, an overview of the various semantic segmentation approaches was discussed that may be suitable for this research. The chosen architectures for this purpose will be discussed in Chapter 3.

Chapter 3

THEORY

This chapter covers the background theory behind Convolutional Neural Networks (CNN). The various layers that form a CNN are covered in detail, along with other important components of the CNN. Lastly, this chapter covers the recent advances in semantic image segmentation techniques and discusses four of the state-of-the-art deep neural architectures used for segmentation.

3.1 Convolutional Neural Network (CNN)

Convolutional Neural networks are an extension of neural networks comprising of neurons with learnable weights and biases. CNNs are designed to be used explicitly with images. This allows us to encode certain properties into the architecture that aid in learning tasks relevant to image data and its distribution. CNNs take advantage of the image as an input and help constrain the architecture and vastly reduce the total number of network parameters. CNNs are made up of a sequence of layers that transform a volume of activations into another using differentiable functions Goodfellow *et al.* (2016). Figure 3.1 shows the basic architecture of a Convolutional Neural network (CNN) with its primary layers. The layers of a CNN and its other core components have been discussed in detail in the following sections (LeCun *et al.* (1998), Lo *et al.* (1995)).

3.1.1 Convolutional Layer

The Convolutional layer is the primary building block that detects low and high-level features in the input with spatial and temporal dependencies. The output volume is a 3-dimensional tensor achieved after using a convolutional kernel or a filter matrix over the

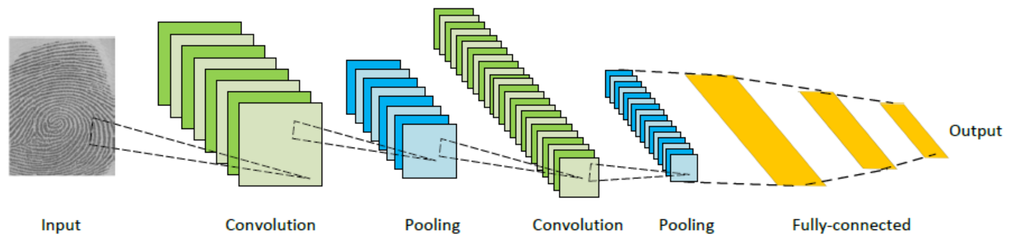


Figure 3.1: Basic Architecture of a Convolutional Neural Network. From LeCun *et al.* (1998)

input image. The resultant output is a feature-map based on the kernel that computes various features from the receptive field of the kernel. The size of the output volume is primarily controlled by three hyper-parameters: depth of the input, stride, size of zero-padding.

1. The depth of the output volume corresponding to the number of filters used to extract various features.
2. The stride value decides the amount of sliding a filter kernel used when performing the convolution operation. The spatial volume of the output reduces as the stride value increases thus reducing the receptive field.
3. To handle the spatial size of the output, it is essential to pad zeros to the input volume around the image channels.

The spatial size of the output volume as described in equation (3.1) is computed using the input volume (W), the size of the filter kernel (F), the stride applied during convolution (S), and the amount of zero-padding used (P) on the input.

$$\text{Spatial size of the Output Volume} = (W - F + 2P)/S + 1 \tag{3.1}$$

For example, an input of size 9×9 with a filter kernel of size 5×5 with a stride of 1 and no zero-padding would give a 7×7 output. If the stride is changed to 2, the output would have the size 3×3.

The convolutional layer generates the output by convolving the filter kernel over the input. The operation can be described as in equation (3.2).

$$y(i,j) = (x * h)[i,j] = \sum_k \sum_l h[k,l] * f[j-k, i-l] \tag{3.2}$$

where:

$y(i,j)$	$=$ Output frame
x	$=$ Input frame
h	$=$ Filter kernel
(i,j)	$=$ Size of the frame
(k,l)	$=$ Size of the Kernel

The total number of parameters for a convolutional layer determines the number of learnable parameters that are updated through training. As the network architecture gets deeper, and more layers are added, the number of learnable parameters increase exponentially and contribute to the model complexity. The equation to estimate the total number of parameters for a layer with an input volume of $W \times H \times D$ is as described in (3.3)

$$\text{Total number of parameters} = (F \times F \times D + 1) \times K \tag{3.3}$$

F represents the size of a filter kernel used. The D in equation (3.3) represents the depth or the number of channels present in the input volume. K represents the number of filter kernels used for convolution. The constant 1 in the equation corresponds to the bias parameter for every filter kernel leading to $(F \times F \times D) \times K$ weights and K biases.

3.1.2 Pooling Layer

The pooling layer typically reduces the spatial size of the representation while extracting dominant features from the data. This aids in reducing the number of parameters in the network thus helping control overfitting. The pooling layer is usually placed right after a convolutional layer as it removes the information and extracts distinct features. It operates on every depth slice of the input volume and uses the MAX operation to resize it spatially. Based on the choice of the pooling method, the size of the filter used to perform the pooling operation varies. The most commonly used form of pooling, also referred to as 'Max Pooling', is using a filter kernel of size 2×2 with a stride of 2 which downsamples the height and width of every channel of the input volume by 2. The depth dimension of the input remains unchanged after pooling. The pooling units can also perform other functions, such as average pooling or even L_2-norm pooling instead of the MAX operation. Max pooling returns the maximum value from the data windowed by the kernel while Average pooling returns the average of all the values in a windowed portion of the image. Max pooling performs better as compared to Average pooling, owing to its de-noising properties and hence, it is used widely in modern architectures Goodfellow *et al.* (2016).

3.1.3 Activation Functions

Activation functions are differentiable functions that introduce a non-linearity to the data passed through it. These functions are attached to every neuron and determine if a neuron will be activated or not based on the output of the function used. As backpropagation is used to train the model parameters, the activation functions need to be computationally efficient. Using non-linear activation functions, the network can learn complex data distributions and aid in providing accurate predictions (Caldelli *et al.* (2019)). The following sections discuss the non-linear activation functions frequently used in modern neural network architectures.

Sigmoid / Logistic

The sigmoid function is expressed as shown in equation (3.4). The sigmoid function has a smooth gradient and it bounds the output values between 0 and 1. It suffers from the vanishing gradient problem for very large or small values and is also computationally expensive. Figure 3.2 shows its graphical representation.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.4}$$

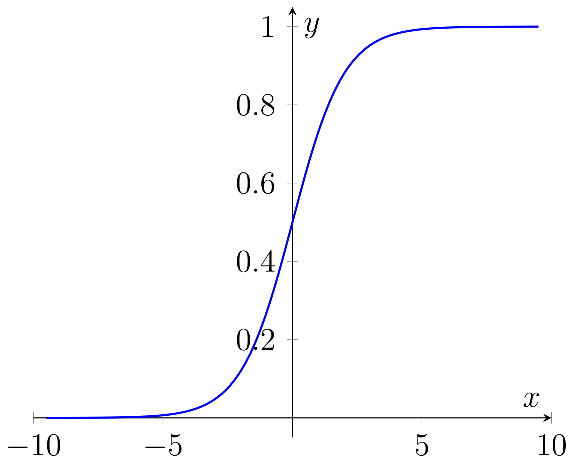


Figure 3.2: Graphical Representation of the Sigmoid Function

TanH / Hyperbolic Tangent

The tanH function is expressed as shown in equation (3.5). The hyperbolic tangent function is very similar to the sigmoid function. It is more zero-centered which helps in modeling inputs that are strongly positive, negative, or neutral. Figure 3.3 shows its graphical representation.

$$\tanh(x) = \frac{e^{-x} - e^{-x}}{e^{-x} + e^{-x}} \tag{3.5}$$

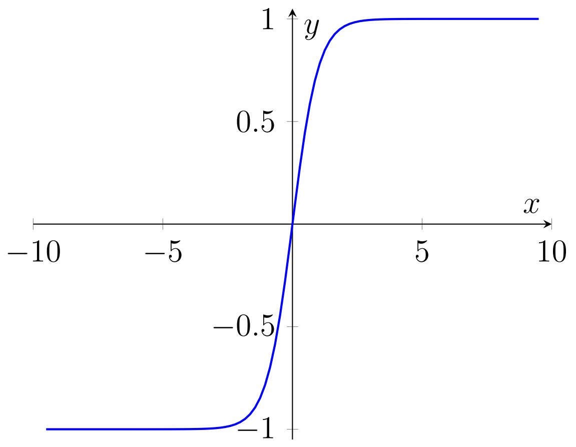


Figure 3.3: Graphical Representation of the tanH Function

Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) function is expressed as shown in equation (3.6). ReLU is more computationally efficient as compared to the sigmoid and the tanH function. Its disadvantage is that it suffers from the Dying ReLU problem in which the gradient of the function becomes zero when the inputs are negative or approach zero. Owing to this, the network cannot perform backpropagation which in turn hinders learning. Figure 3.4 shows its graphical representation.

$$f_{ReLU}(x) = max(0, x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{3.6}$$

Leaky ReLU

The Leaky Rectified Linear Unit (ReLU) function is expressed as shown in equation (3.7). Leaky ReLU solves the dying ReLU problem faced by the ReLU activation function, thus enabling backpropagation for negative values as well. Figure 3.5 shows its graphical representation.

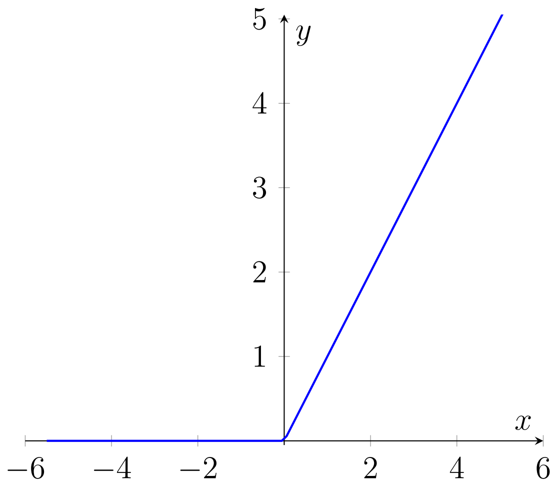


Figure 3.4: Graphical Representation of the ReLU Function

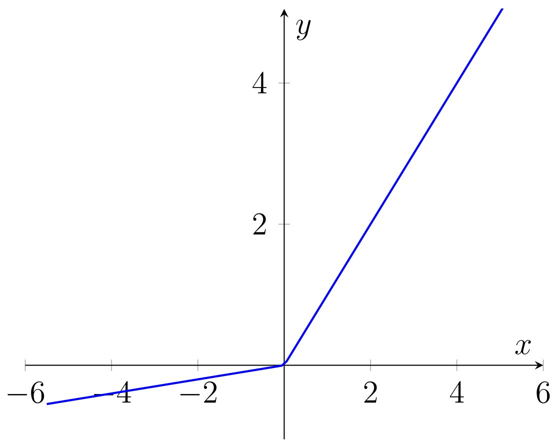


Figure 3.5: Graphical Representation of the Leaky ReLU Function

$$f_{Leaky}(x) = max(0.1 * x, x) = \begin{cases} 0.1x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{3.7}$$

3.1.4 Fully-connected Layer

In a fully-connected layer, all the neurons have complete connections to all activations in the previous layer, each having their weights. The activations are computed using a matrix multiplication followed by a bias offset. The final fully-connected layer is referred to as the 'Output layer' and it represents the class scores or the model output that will be used for prediction or inference. The output layer decided the output size for translating the data propagated through the CNN layers. Based on the desired output, different activation functions may be used in the output layer.

3.1.5 Batch Normalization Layer

Batch Normalization (BN) is an important component of the CNN architecture. This layer normalizes the feature maps by subtracting the mean value of the batch and divides the values by the standard deviation of the batch being processed. This helps in maintaining the feature map distribution by ensuring that the activation does not diverge to very high values. Using BN layers also has regularization effects thus helping prevent overfitting the model.

3.1.6 Dropout Layer

In deep-learning, models are trained upon huge sets of training data. If a model fits very well to a training set with very high training accuracy, it is also able to model the noise present in the training data. This is referred to as *Overfitting*, as the model is not able to generalize or fit observations from the test set on which it has never trained. To avoid overfitting, the Dropout layer is used in DL architectures which randomly chooses a certain portion of neurons and ignores the rest (by not activating them) during the training process. This also leads to improved training speed and prevents overfitting by reducing the number of neurons to train.

3.1.7 Backpropagation

After the input signal propagates through the neural network to the output layer, the parameters (weights and biases) of all the trainable layers need to be updated. Backpropagation is essentially the technique using which a neural network updates the parameters based on the error rates of the output, by propagating the error, back through all the weights using a loss function. The chain rule is used to compute the gradients for every parameter, which are then used to update every parameter for the next iteration. The important part here is to minimize the loss for better training as this minimizes the error difference between the input and the ground truth. Optimizers are used to aid the network in updating its parameters throughout the layers. The Adam optimizer and the Stochastic Gradient Descent (SGD) optimizer are the most commonly used optimizers for training. Adam uses a set of different learning rates that help in achieving convergence faster than SGD which uses a single learning rate (Hecht-Nielsen (1992)).

3.1.8 Transfer Learning

Convolution neural networks have a very high number of trainable parameters viz, weights, and biases. Based on the architecture used, the number of trainable parameters can greatly vary for deep-learning architectures. The weights are updated based on the learning by minimizing the loss to find an optimal value that gives the best possible output. Weights can learn different data features ranging from edge and color information to complex structures and distributions. As mentioned by Goodfellow *et al.* (2016), it is possible to exploit learning's from one setting to generalize in another setting. It is possible to transfer the learning's (viz. weights and biases) of a trained model to another model expected to function on a similar task. Also, model architectures trained from scratch, have random weights set throughout the architecture. Depending upon the model and several other factors, it may take the model a significant amount of time to converge and start learning with a high training accuracy. Using Transfer learning, it is possible to use pre-trained weights to initialize the model with the best possible learned parameters that will increase the throughput and model performance.

3.2 Semantic Image Segmentation

Semantic image segmentation is the process of classifying each pixel in an image belonging to a particular class. Segmentation involves classification and localization of the features. Classification involves making a prediction for the complete input. Localization provides us with the desired classes and the spatial location of those classes. Using dense predictions, labels are inferred for every pixel with the class of its enclosing region. Deep learning architectures have been used specifically for the task of Semantic segmentation and have been able to achieve significant performance boosts in terms of accuracy with the latest advances in deep-learning (Zhou *et al.* (2019a)). The state-of-the-art deep learning architectures typically used for Semantic segmentation tasks of different types of images have different aspects and design elements in common. The encoder-decoder architecture setup, skip connections, Multi-scale and Pyramid-based networks, Dilated Convolutional models are commonly used in many of the State-of-the-art (SOTA) architectures that have achieved the best performance over benchmarking image datasets (Minaee *et al.* (2020)).

3.3 State-of-the-art (SOTA) Architectures

This section discusses some of the advanced deep-learning architectures that have been used for the task of semantic segmentation and have also contributed to the research in this field by introducing advanced techniques and design patterns to improve the segmentation score.

3.3.1 U-Net

The U-Net architecture as discussed by Ronneberger *et al.* (2015) is an encoder-decoder type of deep-learning network architecture initially designed for medical image segmentation. The U-Net comprises of two parts, the encoder architecture that reduces the spatial size of the input to capture the context and the symmetric decoder architecture that aids in precise localization. The down-sampling portion comprises an architecture similar to the Fully Convolution Network (FCN) which extracts image features using 3×3 convolutions. The up-sampling part of the architecture uses up-convolutions to reduce the number of feature maps while increasing the dimensions. It uses 1×1 convolutions at the end to generate a segmentation map categorizing each pixel by processing the feature maps. The upsampling portion of the network has a large number of feature channels, which allows the context information to propagate to higher resolution layers. The U-Net was specifically designed to train with a fewer number of training images to yield precise segmentation results (Lee *et al.* (2017)).

The network architecture as shown in figure 3.6 consists of repeated 3×3 convolutions, followed by a Rectified Linear Unit (ReLU) and a 2×2 max pooling operation for downsampling. At every downsampling step, the number of feature channels is doubled. For upsampling, 2×2 convolution (up-convolution) is used that halves the number of feature channels provided as input to that layer. It also includes skip connections concatenating the output from the encoder architecture (downsampling), followed by two 3×3 convolutions and a ReLU layer. These skip connections provide the localization information from the downsampling portion to the upsampling portion of the architecture. U-Net architecture uses an overlap-tile strategy for training and prediction. The output size is smaller than the input size because of the use of unpadded convolutions. The input image is broken down into patches of a size acceptable to the model input. The model predicts the segmen-

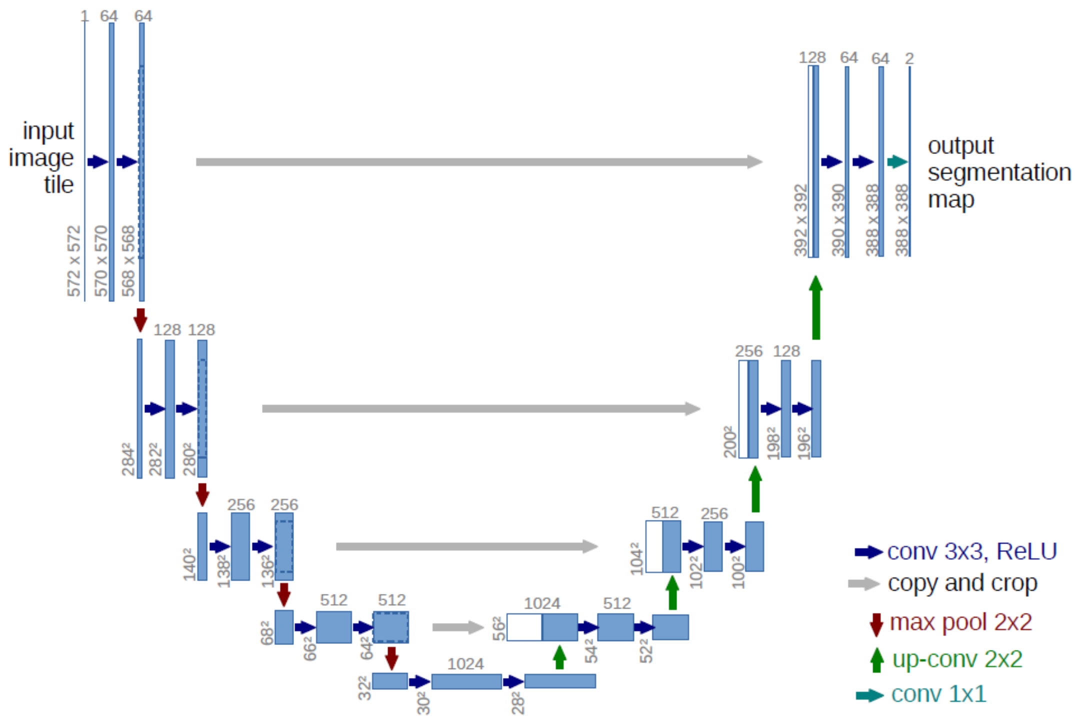


Figure 3.6: U-Net Architecture. From Ronneberger *et al.* (2015)

tation map for every patch and these predictions are then combined to form the resultant segmentation output. U-Net also performs well with data augmentation as it can generalize well to random deformations applied to the input image and the corresponding output segmentation map (Falk *et al.* (2019)).

As discussed in Ronneberger *et al.* (2015), the U-Net implementation also includes a weight map that is applied to the output of the network which is helpful in the separation of class boundaries touching each other. The weight map is computed as shown in equation (3.8)

$$w(x) = w_c(x) + w_0 \cdot \exp\left(-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}\right) \tag{3.8}$$

where $w_c : \Omega \to \mathbb{R}$ is the weight map to balance the class frequencies, $d_1 : \Omega \to \mathbb{R}$ denotes the distance to the border of the nearest cell and $d_2 : \Omega \to \mathbb{R}$ the distance to the border of the second nearest cell. The cross-entropy function is penalized at each pixel position by

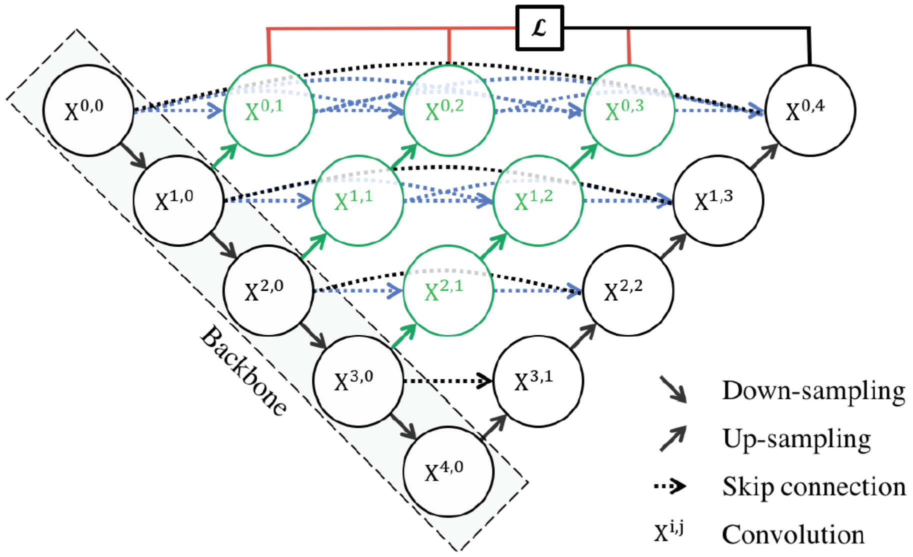


Figure 3.7: U-Net++ Architecture. From Zhou *et al.* (2018)

the weight map and can be expressed as shown in equation (3.9)

$$E = \sum_{x \in \Omega} w(x) \log(p_{l(x)}(x)) \tag{3.9}$$

where $\ell : \Omega \rightarrow \{1, \ldots, K\}$ is the true label of each pixel and $w : \Omega \rightarrow \mathbb{R}$ is the weight map introduced during the training.

3.3.2 U-Net++

The U-Net++ is a nested U-Net architecture developed by Zhou *et al.* (2018) based on nested and dense skip connections for more accurate semantic segmentation. It was designed for the task of image segmentation in the medical domain, yielding an appreciable performance gain over the U-Net architecture.

The architecture of U-Net++ as shown in figure 3.7 consists of an encoder and decoder that are connected through a series of nested dense convolutional blocks. As compared to the U-Net, it bridges the feature maps from the encoder to the decoder before merging them. For instance, in figure 3.7, the semantic gap between $X^{0,0}$ and $X^{1,3}$ are bridged using three intermediate convolutional layers. The components in black color correspond

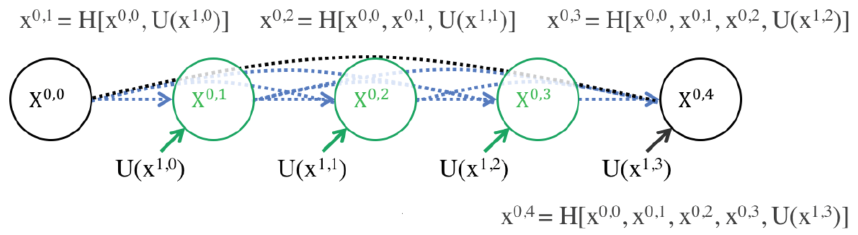

x0,1 = H[x0,0, U(x1,0)]
x0,2 = H[x0,0, x0,1, U(x1,1)]
x0,3 = H[x0,0, x0,1, x0,2, U(x1,2)]
X0,0
X0,1
X0,2
X0,3
X0,4
U(x1,0)
U(x1,1)
U(x1,2)
U(x1,3)
x0,4 = H[x0,0, x0,1, x0,2, x0,3, U(x1,3)]

Figure 3.8: Top-most Skip Pathway of the U-Net++ Architecture. From Zhou *et al.* (2018)

to the original U-Net implementation with the encoder-decoder structure. The dense convolutional blocks on the skip pathways that are used to bridge the semantic gap have been shown in blue and green colors. The portion in red color corresponds to the deep supervision mode used in the U-Net++ architecture. The primary differences that distinguish the U-Net++ from the basic U-Net are the use of re-designed skip pathways connecting two sub-networks and deep supervision (Zhou *et al.* (2019c)).

Re-designed Skip Pathways

The re-designed skip pathways in the U-Net++ architecture introduce dense convolution blocks with a different number of convolutional layers based on the pyramid level. A concatenation layer is used before the convolutional layer which fuses the output from the previous convolutional layer of the dense block along with the up-sampled feature map from the lower dense block.

These dense convolution blocks help in bringing the semantic maps of the encoder closer to the decoder feature maps, thus making the optimization process faster.

The skip pathways can be formulated as shown below in equation (3.10).

$$x^{i,j} = \begin{cases} \mathcal{H}(x^{i-1,j}), & j = 0 \\ \mathcal{H}\left(\left[\left[x^{i,k}\right]_{k=0}^{j-1}, \mathcal{U}(x^{i+1,j-1})\right]\right), & j > 0 \end{cases} \tag{3.10}$$

In equation (3.10), $x^{i,j}$ denotes the output from the node $X^{i,j}$ where the index i refers to the encoder layer and the index j refers to the dense convolutional block on the skip pathway. The $\mathcal{H}$ represents the convolution operation followed by an activation function, $\mathcal{U}$ denotes an upsampling layer, and [] represents the concatenation layer.

Deep Supervision

UNet++ generates full-resolution feature maps at various levels as it uses nested skip pathways between the encoder-decoder parts. Using Deep supervision enables the model to operate in different modes. The outputs from all segmentation branches are averaged in the Accurate mode. The fast mode lends a speed gain in which the final segmentation map is obtained from only one of the segmentation branches. UNet++ uses a combination of binary cross-entropy and dice coefficient as the loss function as shown in equation (3.11)

$$\mathcal{L}(Y, \hat{Y'}) = -\frac{1}{N}\sum_{b=1}^{N}\left(\frac{1}{2}\cdot Y_b \cdot \log \hat{Y_b} + \frac{2 \cdot Y_b \cdot \hat{Y_b}}{Y_b + \hat{Y_b}}\right) \tag{3.11}$$

The $\hat{Y_b}$ denotes the flattened predicted probabilities and Y_b denotes the flattened ground truths of b^{th} image respectively. N indicates the batch size used.

3.3.3 Pyramid Scene Parsing Network (PSPNet)

Pyramid Scene Parsing network (PSPNet) by Zhao *et al.* (2017) is a scene parsing network that makes the use of the Pyramid Pooling module with different-region-based-context for achieving state-of-the-art segmentation performance. The pyramid pooling module provides global information about the images leading to better segmentation results.

Figure 3.9 shows the PSPNet architecture with the Pyramid Pooling module that takes the feature map from the last convolutional layer of the CNN as input. It fuses features under four different pyramid scales. the topmost coarse level in the Pooling module performs global pooling to generate a single output. The pyramid levels separate the feature maps

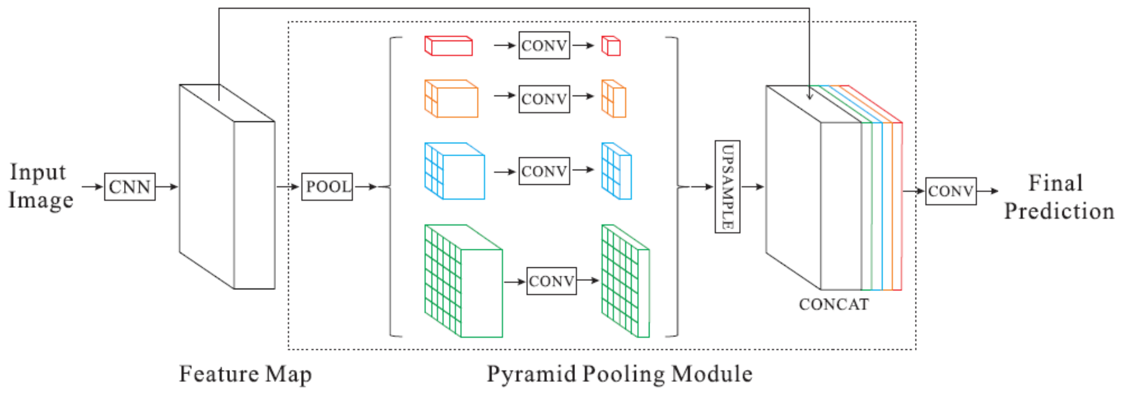
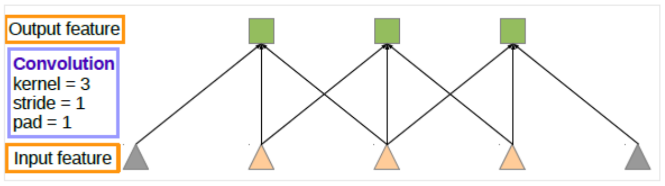


Figure 3.9: PSPNet Architecture. From Zhao *et al.* (2017)

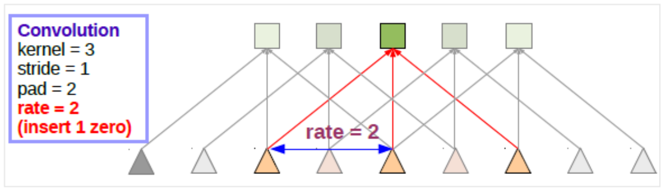
into different sub-regions and form pooled representations of the map for different locations with different sizes. Using 1×1 convolutional layer, the dimensions of the context representation is reduced from N to $1/N$ (Fang and Lafarge (2019)). The low dimension feature maps are upsampled to the original input image size and then concatenated with the original feature map. The pyramid module in the PSPNet as shown in figure 3.9 has four bins of sizes $1 \times 1, 2 \times 2, 3 \times 3$ and 6×6. A pre-trained ResNet model with the dilated network strategy is used to extract the feature map, provided as input to the Pyramid Pooling module. The concatenated output from the pyramid pooling module is then passed through a convolutional layer to generate the prediction map as the output of the PSPNet.

3.3.4 DeepLab v3+

Deeplab v3+ is a deep learning-based semantic image segmentation model presented by Chen *et al.* (2018). The model is based on two main techniques; Atrous convolutions, Atrous Spatial Pyramid Pooling (ASPP), and the use of an encoder-decoder architecture. The above-mentioned techniques paired with an adapted Xception model as a backbone, have resulted in breakthrough mIoU (Mean Intersection over Union) scores on benchmarking testing datasets.



(a) Sparse feature extraction



(b) Dense feature extraction

Figure 3.10: Atrous Convolutions. From Chen *et al.* (2018)

Atrous Convolutions

Atrous convolutions also referred to as dilated convolutions in some research papers, refer to convolution operations in which the stride value used to sample the input is greater than one. This leads to an output with a larger feature map thus leading to enlargement of the field of view of the filters. It is an efficient technique to control the field-of-view of the output thus finding the best trade-off between accuracy and efficiency. Figure 3.10 provides an illustration of sparse and dense feature extraction with normal and atrous convolutions (Chen *et al.* (2014)).

Consider two-dimensional signals, for each location i on the output y and a filter w, atrous convolution applied over the input x can be expressed as shown in equation (3.12).

$$y[i] = \sum_{k} x[i + r \cdot k]w[k] \tag{3.12}$$

where the atrous rate r corresponds to the stride value.

Atrous Spatial Pyramid Pooling (ASPP)

ASPP is an atrous version of the pyramid pooling technique described in the PSPNet architecture in section . In ASPP, the atrous convolutions with different sampling rates are

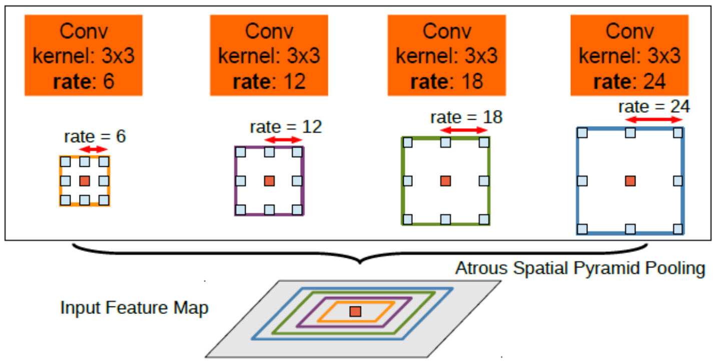


Figure 3.11: Atrous Spatial Pyramid Pooling (ASPP). From Chen *et al.* (2018)

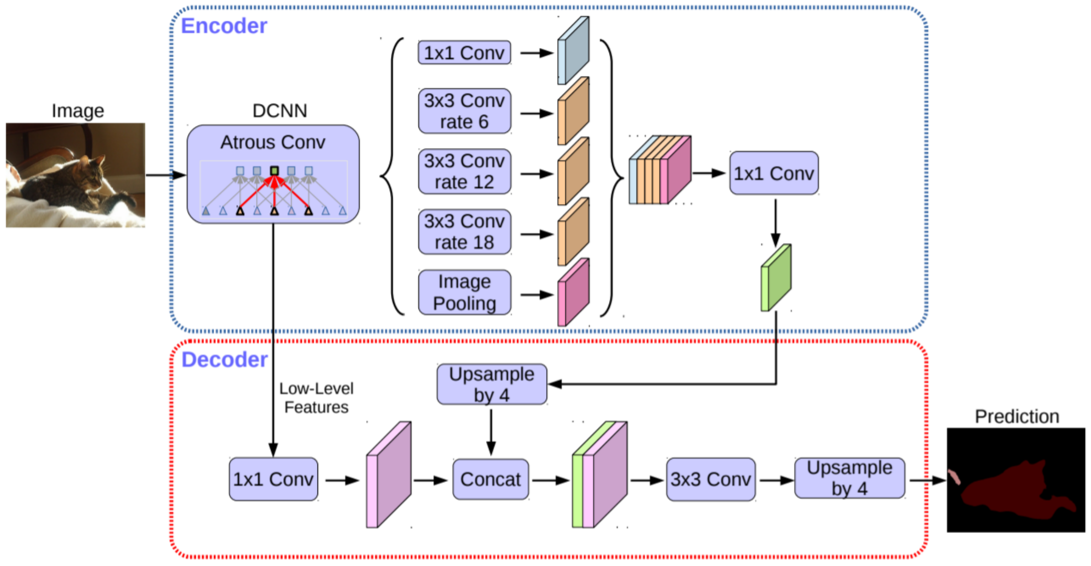


Figure 3.12: Model Architecture of DeepLab v3+. From Chen *et al.* (2018)

computed in parallel and then fused. This helps to account for different scales of the objects in the input classes, thus improving the accuracy. Figure 3.11 shows the multiple filters with the different sampling rates in the ASPP module (Chen *et al.* (2017)).

As shown in figure 3.12, the image features extracted using atrous convolutions are provided to the ASPP which generates the output of the encoder part of the network. The concatenated feature maps from the ASPP are provided to the decoder part through a 1×1 convolution followed by upsampling. This is concatenated with the low-level features received from the first two layers to generate the predicted output.

Chapter 4

EXPERIMENTS, RESULTS AND ANALYSIS

This chapter introduces the dataset containing the Aluminum Alloy 7075 images used for semantic segmentation, the methodology used for data generation, pre-processing, and other details regarding its usage. It then covers the various experiments conducted with the baseline and different state-of-the-art architectures and relevant modifications made during their implementations. It also covers a comparative analysis of the results obtained after conducting these experiments and a performance analysis based on the Training Time, Training complexity, GPU resources used, and scores obtained.

4.1 Dataset

4.1.1 Data Generation Process Using X-ray Microscopy

Aluminum alloy AA 7075 is a complex heterogeneous alloy consisting of particles such as metallic inclusions and precipitates that affect the mechanical and corrosion behavior of the alloy. In this study, Transmission X-ray Microscopy (TXM) has been employed to non-destructively study the microstructure and corrosion behavior of Aluminum Alloy 7075 in 4D. Micropillars of AA7075 were fabricated using a Ga+ focused ion beam as a part of a Zeiss Auriga Scanning Electron Microscope at Arizona State University. Transmission X-ray microscopy was conducted on the pillars at the 32-ID-C beamline at the Advance photon source at Argonne National Laboratory. The combination of a condenser lens, custom-made Fresnel zone plates, and a camera binning value of 1, yielded a pixel size of 18 nm for this experiment. Further details about the beamline can be found in De Andrade *et al.* (2016) and Kaira *et al.* (2017). X-ray energy of 9.75 keV was used during the experiment as it was above the Zn absorption edge. This ensured a good contrast between the precipitates and inclusions. Each scan involved capturing 1200 projections over an angular range of 0-180 degrees with an exposure time of 1 second/ projection, thereby making the duration of each scan to be 20 minutes. The datasets for both the peak aged and overaged specimens were reconstructed using TomoPy, a Python-based toolbox used to reconstruct and analyze synchrotron tomography-based data. A filtered back-projection algorithm was used to reconstruct the projections into a reconstructed microstructure. The datasets were then cropped appropriately, and a set of filters were applied to reduce noise and sharpen the data on Avizo 9 (Bethesda, MD). Non-local means were employed to reduce noise in the data and an unsharp mask was used to sharpen the relevant features in the images, such as the inclusion and precipitate particles. Avizo 9 was then further used to perform image segmentation on these datasets to visualize and quantify the distribution of precipitates, inclusions, and the corrosion damage.

4.1.2 Aluminum 7075 Data

The dataset used for this research comprises of images of metallic inclusions and precipitates over an Aluminum substrate over time. The data generation process for the raw and segmented images is as per section 4.1.1. The images have 3 classes namely the Precipitates, Inclusions, and the Al Alloy substrate. The current dataset consists of 1024 raw and segmented image pairs referred to as the ‘Aluminum 7075 dataset’ in this report. The raw images in the dataset have a size of 792×774 are in the .tif (Tagged Image File Format) with a high bit-depth of 32 bits. Figure 4.1 shows a sample of raw and ground-truth segmented images from the dataset. The segmented images have a bit depth of 8 bits described in 3 color values representing the 3 classes (including the background class). The raw data in the images need to be pre-processed and normalized before training to avoid overfitting during training. Based on the model architecture used, the segmented image data was also

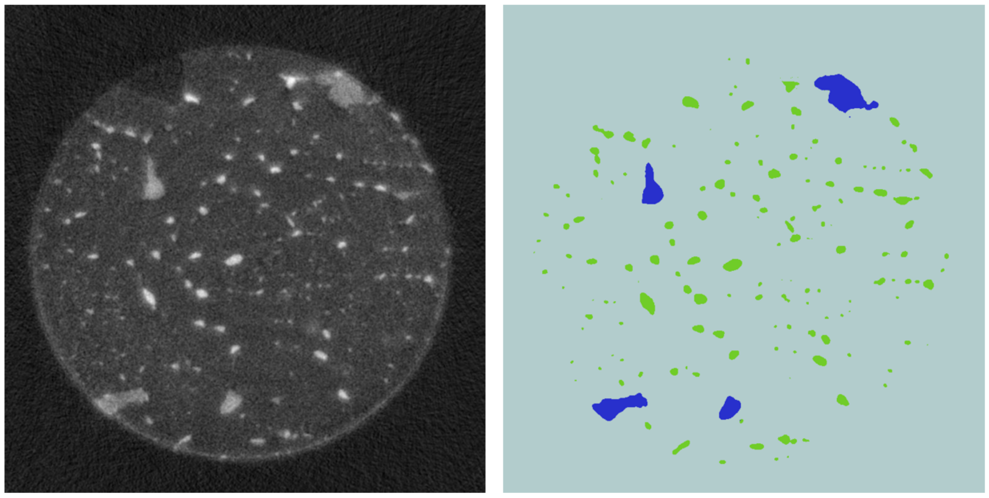


Figure 4.1: Sample Raw and Segmented Image from the AA 7075 Dataset

converted into a One-hot encoding format or Label encoding format. The Pre-processing techniques and Data augmentation methods used have been covered in the next section.

4.1.3 Training, Validation and Testing Set

The Aluminum 7075 Dataset contains 1024 images pairs out of which 184 images have missing precipitates and inclusions. These images are the specimens captured at the beginning and end of the corrosion study on the Aluminum 7075 Alloy using X-Ray Microscopy. For training and testing, only the remaining 840 images have been considered to handle data imbalance for the classes present. In machine-learning and deep-learning applications, it is ideal to split the dataset into a Training and a Testing set. In some cases, I have also used a validation set by randomly selecting up to 10 images from the training set to validate the accuracy and other metrics during training epochs. I have used an 80-20 split of the dataset viz. 80 percent of the images for training (620 images) and 20 percent of the images for testing (200 images). The models are trained on the training data pairs and the predictions or model evaluations are done by using the testing set. The data pairs used for training include the raw image and the segmented output based on which the model learns to generalize on other data.

4.1.4 Data Pre-processing and Augmentation

The image data in the raw unsegmented training images have a high bit-depth and can greatly affect a model's capacity to fit to a training set whilst affecting its learning ability. I have also applied Mean normalization to normalize the images before the training, This allows scaling the image data in the range of zero to one, which can then be scaled to the JPG image range by multiplying it by 255. Model architectures like the DeepLab v3+ train specifically on images with 3 channels (viz. RGB). In this case, a single channel comprising the raw data was repeated for each channel, thus forming the 3-channel RGB image.

Deep Neural Networks perform best when trained on larger sets of data. Convolutional Neural Networks incorporating various layers contributing to the deep architecture often fit well to the training set when fed with a huge dataset of images. Using data augmentation, it is possible to artificially create modified versions of the images in the dataset which can be used to training. This helps in increasing the size of the training set, to an amount suitable for deep-learning techniques, while improving a model's performance and its ability to generalize. I have used the 'ImageDataGenerator' class provided by Keras to perform augmentations on the training set pairs. The image transforms done during data augmentation include image manipulation techniques such as shifts, flips, zooms, rotations, brightness variations, etc. I have used a random set of these operations while training to augment up to 5-6 pairs of images per image in the training set.

4.2 Metrics for Model Evaluation

I have adopted the most commonly used metric viz. Intersection over Union (IoU), in the field of Semantic image segmentation, to compare the performance of various segmentation-based methods.

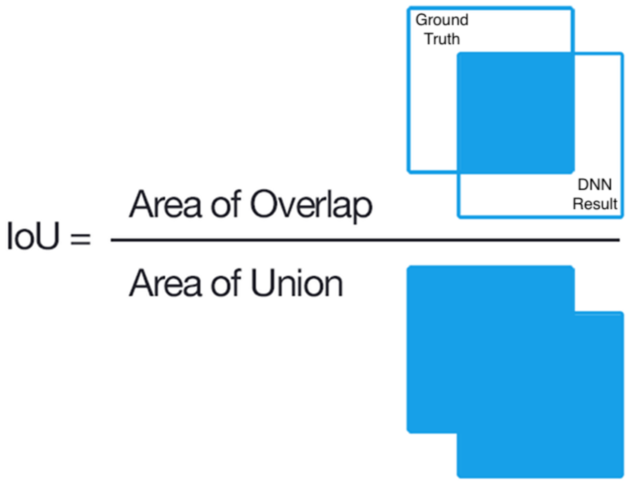


Figure 4.2: Intersection over Union (IoU)

4.2.1 Intersection over Union (IoU)

The Intersection over Union for a semantic class m, given the ground-truth mask and predicted binary mask, can be defined as:

$$IoU_m = \frac{t_{p_m}}{t_{p_m} + f_{p_m} + f_{n_m}} \tag{4.1}$$

In the equation above, t_{p_m} refers to the number of true-positives, f_{p_m} refers to the number of false-positives and f_{n_m} refers to the number of false-negatives in the predicted masks. IoU can also be expressed as the ratio of the area of overlap between the predicted mask and the ground truth divided by the area of union between the predicted mask and the ground truth, as shown in figure 4.2.

The Mean IoU (mIoU) score is commonly used to compare different methods employing Semantic segmentation by finding the mean of the IoU scores for every class present in the output. The IoU metric tends to penalize single instances of misclassification in the predicted output. Owing to this, the IoU metric measures performance closer to the worst-case scenario.

4.3 Baseline U-Net Results for Different Experiments

The baseline U-Net used for initial experimentation was obtained from the Github repository XLearn (Tomography), a deep-learning toolbox for X-ray imaging (Kaira *et al.* (2018), Yang *et al.* (2017), Yang *et al.* (2018)). It is a basic U-Net that performs training and inference by doing patch-by-patch processing on the training/testing images. The existing model only trained upon one image for up to 50 epochs before generating the predicted image. In this case, the model performed binary segmentation and overfit to the single image used for training and was unable to generalize to the varying distributions from the various images in the training set. I trained the model on a single image and a set of 10 images enough for the GPU to train upon and recorded the mIoU results obtained on 5 testing images as shown in table 4.1. The results obtained by training the model upon a single input image with a mean IoU score of 62.26 is referred to as the Baseline result for the scope of this research.

To begin with, I modified the process of loading the images from the training and testing dataset by making use of mini-batches. The U-Net converts every image input into patches of the provided size (e.g. 32×32). Owing to this, the total number of image patches is very high and it consumes a huge chunk of the memory while processing. With the help of mini-batching, I was able to handle the process of generating the image patches and train upon a fixed batch size of these patches which could easily fit in the allotted GPU memory. The batch size selected for processing is made sure to be exactly divisible by the number of GPU resources used. Different sets of hyper-parameters to fine-tune the model for the task by varying the patch sizes, size of the batch used for processing the train/test input, learning rate, optimizer, and loss function were tried.

Type	Training Image(s)	Epochs	mIoU Score	Batch size	GPU Resources used (Max. 8)
Loaded without	**1**	**50**	**62.26**	**-**	**2**
Mini-batching	10	30	60.19	-	2
Loaded with	100	5	40.65	128	2
Mini-batching	200	5	44.40	512	4
	620	**5**	**49.40**	**128**	**4**

Table 4.1: Baseline U-Net Results for Binary Segmentation on 5 Testing Images with and without Mini-batching

I added the Batch normalization layer to the baseline U-Net implementation, to allow the model to effectively fit by normalizing the feature input before every ReLU activation layer. Based on the initial few epochs for a small subset of the training set, the best set of hyper-parameters were decided before initiating the training process.

The baseline model I used, only performed binary-segmentation on the input images. The training input to the model and the last layer output (using Softmax activation to change the number of output classes) were modified to generate the model output for 3 classes viz. the inclusions, precipitates, and the background class. The model was trained on all 620 training images with a patch size of 32×32, the Adam optimizer with the MSE loss (binary segmentation), and the Cross-entropy loss (3-class segmentation). The results for binary segmentation and Multi-class (3-class) semantic segmentation have been shown in the table 4.2. The U-Net gave better results in the case of binary segmentation as the precipitates and inclusion were together segmented as one class, the background being another class. In the case of multi-class segmentation, the representation and distribution of the 3 classes in the images varies which results in a comparatively reduced mean IoU score of all the classes.

Segmentation	**Training Images**	**Epochs**	**IoU score**
Binary	**620**	**10**	**74.42**
Multi-class	620	10	73.88

Table 4.2: U-Net Results on the Testing Set for Binary and Multi (3-class) Segmentation

Table 4.3 shows the IoU scores obtained after training the model for multi-class segmentation on different patch sizes for the complete testing set. The patch size 32×32 gives the highest IoU as compared to the other patch sizes as it is the optimum patch size enough to represent the distribution of the particles in the image. After training on multiple patches of different images, the model can generalize well leading to improved performance (Zhang *et al.* (2018)).

Patch size used	**IoU score**
32 × 32	**73.88**
64×64	70.35
128×128	69.16

Table 4.3: U-Net Results on the Testing Set for Different Patch-sizes (Multi-class Segmentation)

4.4 Comparative Analysis of the Results Obtained from State-of-the-art Architectures

The segmentation architectures discussed in section 3.3 were implemented for this research. The PSPNet and UNet++ can train on one-hot encoded segmented ground-truth images. In the case of DeepLab v3+ (based on Tensorflow), I had to convert the complete dataset into Tensorflow's '.tfrecord' format which is a compression format to efficiently store and load training/testing data. The models can train faster on tfrecords as they consume less memory during processing and are loaded faster from memory. The base implementations of the architectures/backbones used for the purpose were referred to from existing open-source implementations.

The training and testing for these DL models were done in Tensorflow/Keras with the use of GPU computational resources. Mean normalization of the training data was done to reduce the computation before training on every image. Based on the number of GPU resources available, during the time of training, a fixed batch size was used and hyper-parameters like learning-rate, weights initialization, and class weights were modified based on a few initial epochs. Depending upon the backbone supported by the implementation, the various architectures were trained and the segmentation results viz. mIoU scores were recorded and are as shown in table 4.4.

The DeepLab v3+ works comparatively better as compared to the U-Net, the U-Net ++, and the PSPNet owing to Atrous Spatial Pyramid Pooling (ASPP) technique. Its encoder-decoder architecture paired with the ASPP module helps in the effective segmentation of images with a different number of classes. It is also able to segment images with overlapping classes as shown in figures 4.3 and 4.4. In the case of DeepLab v3+, I also used Data augmentation techniques to augment newer images for every image supplied as input during the training process. Table 4.4 compares the mIoU scores of the different architectures implemented to segment the inclusions and precipitates from the images of the Aluminum alloy AA 7075. The DeepLab v3+ has an mIoU score of 74.27 with slightly better and consistent segmentation results as shown in figure 4.4. The mIoU scores represent the mean IoU scores of all the 3 classes present in the segmented output. The IoU score for the background class which occupies the major portion of the images is very high, in the range of 88 to 95, while the IoU scores of the other 2 classes range from 55 to 76 depending upon the implementation. I have used class weights during training to highlight the importance of segmenting the precipitates and inclusion classes over the background class.

Architecture	**Variant/Backbone**	**mIoU Score**
U-Net	Baseline	62.26
	Fine-tuned	73.88
U-Net ++	Inception-v3	72.43
	ResNet-152	72.19
PSPNet	Inception-v3	72.36
	ResNet-152	72.80
DeepLab v3+	**Xception**	**74.27**

Table 4.4: Mean IoU Segmentation Scores on the Testing Set Using Different Architectures and Backbones

Figure 4.5 shows the incorrect segmentation output of images predicted from the testing data. *Image 939* has incomplete data and is a part of the set of images captured at the very beginning and end, using X-ray microscopy. The ground-truth only has a few particles at the top (enclosed in the red box) but the models identify the various disturbances in the image and classify them as relevant classes. In the case of *Image 112*, the ground-truth does

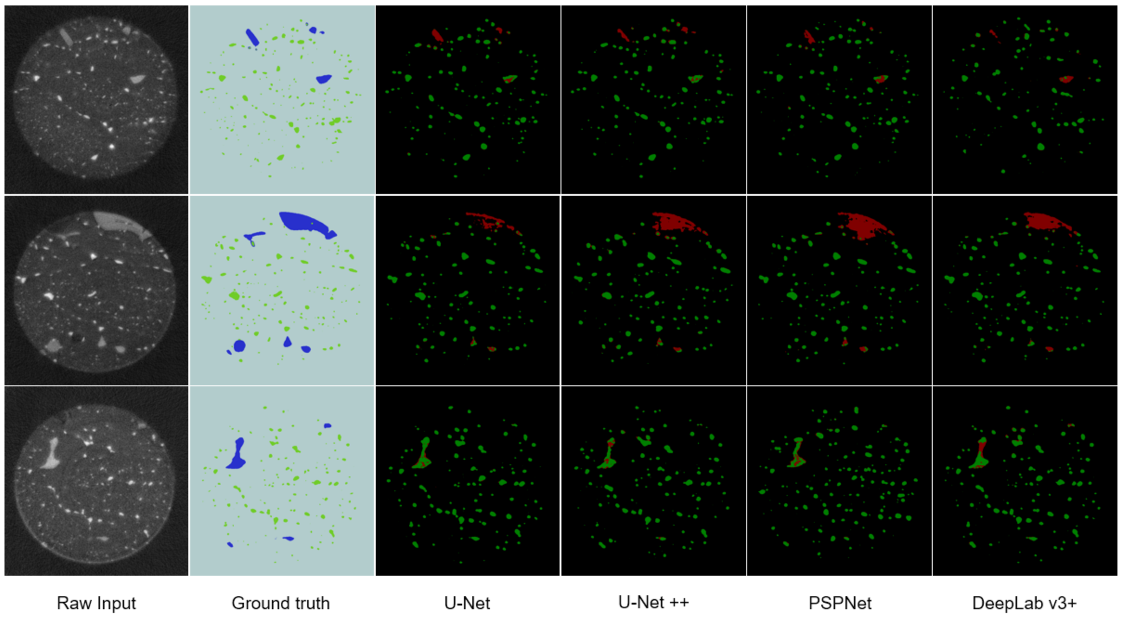


Figure 4.3: Comparison of Raw, Ground-truth and Segmented Output of Images 305, 555 and 699 Using Different DL Architectures

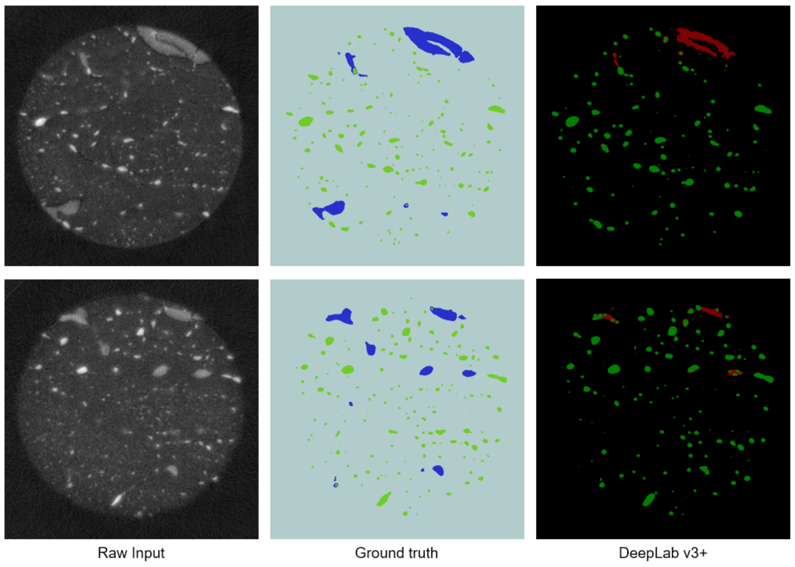


Figure 4.4: Segmented Output of Images 482 and 234 Using DeepLab v3+

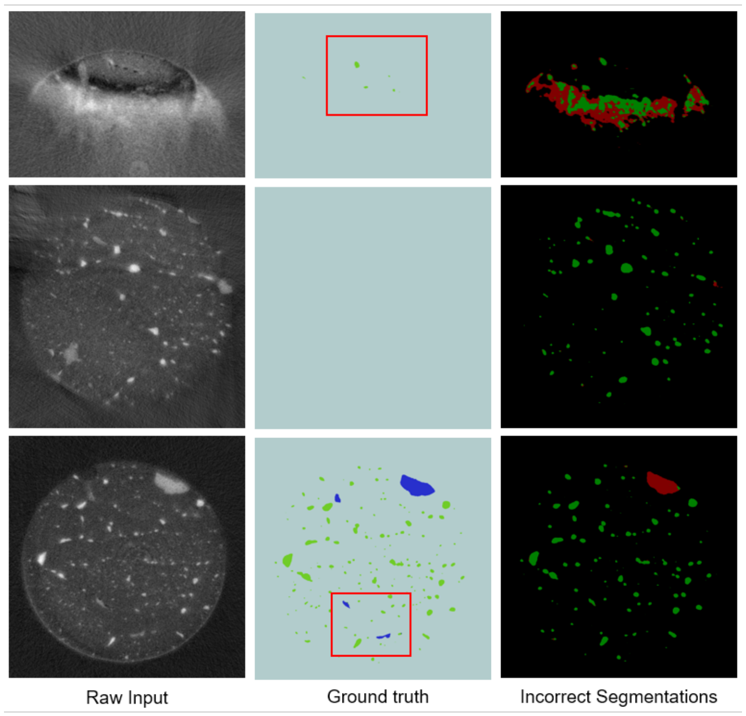


Figure 4.5: Incorrect Segmentation of Raw Input Images 939, 112, 772

not have any class information but the segmented output does identify the precipitates and inclusions present in the input image. To handle such scenarios better, it would be ideal to use data augmentation techniques to create more such data with disturbances, occlusions, etc to help the model generalize to the variability of the classes during prediction. It is also noticed that the model performs better for the class with a higher presence in the image. With the use of class weights, I have also adjusted the values to handle the class imbalance scenario. *Image 772* is a standard input image with a good presence of both the particles. The red box highlights the blue particles present in the ground-truth which were not segmented in the output.

4.5 Performance Analysis of the Models

I conducted a performance analysis to infer which model implementation would be ideal to achieve the highest Mean IoU score for the task of Semantic segmentation with minimal use of GPU resources, requiring lesser time for training/prediction. The various factors that I have used to assess the model with the highest performance are the number of epochs used for training, the batch size, active GPU resources used during training/prediction, the time required to train on the complete training set, the time required to generate prediction images for an image and the resultant Mean IoU score.

Architecture	**Batch Size**	**Epochs**	**Training Time (days)**	**GPU Resources used (max. 8)**	**Prediction Time per Image (secs)**	**mIoU Score**
U-Net	2048	5	4 to 4.5	4	25	73.88
U-Net++	2	10	1 to 1.2	4	30	72.43
PSPNet	2	10	1	4	30	72.36
DeepLab v3+	**8**	**5000**	**0.5**	**2**	**15**	**74.27**

Table 4.5: Performance Analysis of the Model Implementations

Table 4.5 compares the four deep-learning architecture I have implemented based on the factors mentioned above. Here, the U-Net model implementation we used takes the highest amount of time owing to the process of patch-by-patch processing, which adds up to computations and consumes a major chunk of the GPU resources. The U-Net++ and the Pyramid Scene Parsing Network (PSPNet) are able to achieve favorable mean IoU scores in a training period of 1 day, but they require the availability of more GPU resources for smaller batch size. The DeepLab v3+ outperforms the other model implementation in terms of faster training time and IoU scores, as it can train in approximately 12 hours by processing multiple images in parallel while handling an optimal batch size consuming minimal GPU resources. The DeepLab v3+ thus proves to be an ideal fit for training upon the Aluminum AA 7075 images with a mIoU score of 74.27.

Chapter 5

CONCLUSION

In this thesis, I have successfully implemented the four state-of-the-art architectures proven to perform well for Semantic image segmentation in various domains using deep-learning. The experiments I performed on the U-Net baseline model and with the other deep-learning architectures show modest improvement in Mean Intersection-over-Union (mIoU) scores. These experiments also show the importance of mean normalization, mini-batching, choice of architecture, and other factors that contribute to improved semantic segmentation performance. I have been able to achieve favorable performance as compared to the baseline method, consuming less time for training while utilizing minimal GPU resources, using the DeepLab v3+ architecture with pre-trained weights. The comparative analysis also helps reiterate the strengths of different approaches and select an architecture optimized for semantic segmentation of X-ray microscopy images.

5.1 Future Directions

There are several future avenues for research in Semantic segmentation for X-ray microscopy images. In the field of deep-learning, there has been consistent progress and the performance of various models for different Computer Vision (CV) tasks has also increased over time. The state-of-the-art architectures I have implemented for this research have various modified versions catering to different types of data from various domains employing new techniques and modifications to the original architecture (Zhou *et al.* (2019b), Zhou *et al.* (2019c), Li *et al.* (2020a)). There is also room for improvement in the configurations and implementations of the PSPNet and Deeplab v3+ models used for training.

For this task, I have used a training set of 1024 images pair. In the future, it would be ideal to train/test the models on more image data pairs with the help of Data Augmentation, consisting of various corner-case scenarios based on which the models would be able to generalize well to the complications in the data. Using augmentations, it would be possible to considerably reduce the misclassification and incorrect segmentation of the classes in the output, helping the model to generalize well to more variations in the data. Alternatively, it is also possible to employ weakly supervised learning methods for dealing with the problem of insufficiency in high-quality hand-labeled data samples.

Some research work related to Semantic segmentation also points towards the usage of shape-aware models (Navarro *et al.* (2019), Clough *et al.* (2019)) that are aware of the geometric shape of the classes in the segmented output. Models exploiting such properties with the help of shape-aware loss functions (Al Arif *et al.* (2017), Hayder *et al.* (2016)) or feature maps that highlight such information may have a further scope in improving the segmentation performance for the overlapping inclusions and precipitates found in X-ray microscopy images of Aluminum alloy particles (Murugesan *et al.* (2019), Li *et al.* (2020b)).

REFERENCES

Al Arif, S. M. R., K. Knapp and G. Slabaugh, “Shape-aware deep convolutional neural network for vertebrae segmentation”, in “International Workshop and Challenge on Computational Methods and Clinical Applications in Musculoskeletal Imaging”, pp. 12–24 (Springer, 2017).

Caldelli, R., R. Becarelli, F. Carrara, F. Falchi and G. Amato, “Exploiting cnn layer activations to improve adversarial image classification”, in “2019 IEEE International Conference on Image Processing (ICIP)”, pp. 2289–2293 (IEEE, 2019).

Chen, D., D. Guo, S. Liu and F. Liu, “Microstructure instance segmentation from aluminum alloy metallographic image using different loss functions”, Symmetry **12**, 4, 639 (2020).

Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs”, arXiv preprint arXiv:1412.7062 (2014).

Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”, IEEE transactions on pattern analysis and machine intelligence **40**, 4, 834–848 (2017).

Chen, L.-C., Y. Zhu, G. Papandreou, F. Schroff and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation”, in “Proceedings of the European conference on computer vision (ECCV)”, pp. 801–818 (2018).

Ciresan, D., A. Giusti, L. M. Gambardella and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images”, in “Advances in neural information processing systems”, pp. 2843–2851 (2012).

Clough, J. R., I. Oksuz, N. Byrne, V. A. Zimmer, J. A. Schnabel and A. P. King, “A topological loss function for deep-learning based image segmentation using persistent homology”, arXiv preprint arXiv:1910.01877 (2019).

Cohen, A., E. Rivlin, I. Shimshoni and E. Sabo, “Memory based active contour algorithm using pixel-level classified images for colon crypt segmentation”, Computerized Medical Imaging and Graphics **43**, 150–164 (2015).

De Andrade, V., A. Deriy, M. J. Wojcik, D. Gürsoy, D. Shu, K. Fezzaa and F. De Carlo, “Nanoscale 3d imaging at the advanced photon source”, SPIE Newsroom **10**, 2.1201604, 006461 (2016).

Falk, T., D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald *et al.*, “U-net: deep learning for cell counting, detection, and morphometry”, Nature methods **16**, 1, 67–70 (2019).

Fang, H. and F. Lafarge, “Pyramid scene parsing network in 3d: Improving semantic segmentation of point clouds with multi-scale contextual information”, ISPRS Journal of Photogrammetry and Remote Sensing **154**, 246–258 (2019).

Goodfellow, I., Y. Bengio, A. Courville and Y. Bengio, *Deep learning*, vol. 1 (MIT press Cambridge, 2016).

Gupta, L. and T. Sortrakul, "A gaussian-mixture-based image segmentation algorithm", Pattern Recognition **31**, 3, 315–325 (1998).

Hayder, Z., X. He and M. Salzmann, "Shape-aware instance segmentation", arXiv preprint arXiv:1612.03129 **2**, 5, 7 (2016).

Hecht-Nielsen, R., "Theory of the backpropagation neural network", in "Neural networks for perception", pp. 65–93 (Elsevier, 1992).

Ikeuchi, K., *Computer vision: A reference guide* (Springer Publishing Company, Incorporated, 2014).

Kaira, C. S., V. De Andrade, S. S. Singh, C. Kantzos, A. Kirubanandham, F. De Carlo and N. Chawla, "Probing novel microstructural evolution mechanisms in aluminum alloys using 4d nanoscale characterization", Advanced Materials **29**, 41, 1703482 (2017).

Kaira, C. S., X. Yang, V. De Andrade, F. De Carlo, W. Scullin, D. Gursoy and N. Chawla, "Automated correlative segmentation of large transmission x-ray microscopy (txm) tomograms using deep learning", Materials Characterization **142**, 203–210 (2018).

Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in "Advances in neural information processing systems", pp. 1097–1105 (2012).

LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE **86**, 11, 2278–2324 (1998).

Lee, D., J. Yoo and J. C. Ye, "Deep residual learning for compressed sensing mri", in "2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)", pp. 15–18 (IEEE, 2017).

Li, M., D. Chen, S. Liu and F. Liu, "Grain boundary detection and second phase segmentation based on multi-task learning and generative adversarial network", Measurement p. 107857 (2020a).

Li, S., C. Zhang and X. He, "Shape-aware semi-supervised 3d semantic segmentation for medical images", arXiv preprint arXiv:2007.10732 (2020b).

Liu, X., Z. Deng and Y. Yang, "Recent progress in semantic image segmentation", Artificial Intelligence Review **52**, 2, 1089–1106 (2019).

Lo, S.-C. B., H.-P. Chan, J.-S. Lin, H. Li, M. T. Freedman and S. K. Mun, "Artificial convolution neural network for medical image pattern recognition", Neural networks **8**, 7-8, 1201–1214 (1995).

Ma, B., X. Ban, H. Huang, Y. Chen, W. Liu and Y. Zhi, "Deep learning-based image segmentation for al-la alloy microscopic images", Symmetry **10**, 4, 107 (2018).

Maldonado-Bascón, S., S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gómez-Moreno and F. López-Ferreras, "Road-sign detection and recognition based on support vector machines", IEEE transactions on intelligent transportation systems **8**, 2, 264–278 (2007).

Minaee, S., Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz and D. Terzopoulos, "Image segmentation using deep learning: A survey", arXiv preprint arXiv:2001.05566 (2020).

Moon, N., E. Bullitt, K. Van Leemput and G. Gerig, "Automatic brain and tumor segmentation", in "International Conference on Medical Image Computing and Computer-Assisted Intervention", pp. 372–379 (Springer, 2002).

Murugesan, B., K. Sarveswaran, S. M. Shankaranarayana, K. Ram, J. Joseph and M. Sivaprakasam, "Psi-net: Shape and boundary aware joint multi-task deep network for medical image segmentation", in "2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)", pp. 7223–7226 (IEEE, 2019).

Navarro, F., S. Shit, I. Ezhov, J. Paetzold, A. Gafita, J. C. Peeken, S. E. Combs and B. H. Menze, "Shape-aware complementary-task learning for multi-organ segmentation", in "International Workshop on Machine Learning in Medical Imaging", pp. 620–627 (Springer, 2019).

Ronneberger, O., P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in "International Conference on Medical image computing and computer-assisted intervention", pp. 234–241 (Springer, 2015).

Seyedhosseini, M., M. Sajjadi and T. Tasdizen, "Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks", in "Proceedings of the IEEE international conference on computer vision", pp. 2168–2175 (2013).

Wang, W., "Rock particle image segmentation and systems", Pattern recognition techniques, technology and applications pp. 197–226 (2008).

Yang, X., V. De Andrade, W. Scullin, E. L. Dyer, N. Kasthuri, F. De Carlo and D. Gürsoy, "Low-dose x-ray tomography through a deep convolutional neural network", Scientific reports **8**, 1, 1–13 (2018).

Yang, X., F. De Carlo, C. Phatak and D. Gürsoy, "A convolutional neural network approach to calibrating the rotation axis for x-ray computed tomography", Journal of Synchrotron Radiation **24**, 2, 469–475 (2017).

Zhang, Z., Q. Liu and Y. Wang, "Road extraction by deep residual u-net", IEEE Geoscience and Remote Sensing Letters **15**, 5, 749–753 (2018).

Zhao, F. and X. Xie, "An overview of interactive medical image segmentation", Annals of the BMVA **2013**, 7, 1–22 (2013).

Zhao, H., J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid scene parsing network", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 2881–2890 (2017).

Zhou, B., H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso and A. Torralba, "Semantic understanding of scenes through the ade20k dataset", International Journal of Computer Vision **127**, 3, 302–321 (2019a).

Zhou, Y., W. Huang, P. Dong, Y. Xia and S. Wang, "D-unet: a dimension-fusion u shape network for chronic stroke lesion segmentation", IEEE/ACM transactions on computational biology and bioinformatics (2019b).

Zhou, Z., M. M. R. Siddiquee, N. Tajbakhsh and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation", in "Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support", pp. 3–11 (Springer, 2018).

Zhou, Z., M. M. R. Siddiquee, N. Tajbakhsh and J. Liang, "Unet++: Redesigning skip connections to exploit multiscale features in image segmentation", IEEE transactions on medical imaging **39**, 6, 1856–1867 (2019c).