ICIC Express Letters
Part B: Applications
Volume **12**, Number **12**, December **2021**

 ISSN 2185-2766
pp. **1169–1176**

DEEP LEARNING-BASED WELD SPOT SEGMENTATION USING MODIFIED UNET WITH VARIOUS CONVOLUTIONAL BLOCKS

Oskar Natan*, Diyah Utami Kusumaning Putri and Andi Dharmawan

Department of Computer Science and Electronics
Faculty of Mathematics and Natural Sciences
Universitas Gadjah Mada
Bulaksumur, Yogyakarta 55281, Indonesia
*Corresponding author: oskarnatan@ugm.ac.id; {diyah.utami.k; andi_dharmawan}@ugm.ac.id

Received February 2021; accepted May 2021

Abstract. *Welding inspection is an absolute need for an industrial factory to ensure the quality of weld joins. However, most of the industry still uses manual inspection which can be subjective, full of bias, and it will lead to inconsistency of the quality standard. Therefore, an intelligent system that can check the quality of welding automatically is needed. As the first step in developing this system, this research aims to create a knowledge model based on deep learning and computer vision that is used to segment weld spots of iron. A Convolutional Neural Network (CNN) model that adopts the architecture of UNet is used as the main model with plenty of modification on its architecture and hyperparameter tuning. The study of using several convolutional blocks is also conducted to achieve the best model configuration. The model works by segmenting the captured image or video frame to determine the region of weld spots. As a result, a modified UNet model with dense convolutional blocks on its bottleneck has the best performance according to the percentage of Intersection over Union (IoU) which reaches 75.45% on the validation set.*

Keywords: Weld spot, Image segmentation, Deep learning, UNet, Convolutional blocks

1. **Introduction.** Welding is one of the important joining technologies in the manufacturing industry. For instance, in a single-vehicle, there are at least 3000 weld spots with different types. A certain kind of inspection technique is needed to ensure the quality of the welding. To date, a visual inspection and evaluation process for detecting welding defects is still manually operated by the human-eye. As a result, it leads to high subjectivity and bias. This mechanism also needs a considerable amount of time and labor which is very inefficient for production. The study of the welding defects detection method has far-reaching significance for ensuring the quality of the product, improving service life, and economic benefits. An approach with an intelligent system based on computer vision can be used to improve the efficiency and accuracy of welding detection. With the rapid development of artificial intelligence, there are various machine learning methods that have been applied in many fields. As a new field of machine learning, deep learning especially CNN indicates great potential for welding detection by continuously reducing the dimension of the features and effectively improving the detection accuracy.

Recently, CNN shows great potential in the field of detection; thus various extensions of CNN models have been developed. For instance, Kumar et al. [1] identify the weld defects using 79 radiographic weld images with 8 defects. The image database has been pre-processed and the features have been extracted with Gray Level Co-occurrence Matrices (GLCM) into 8 and 64 level features and fed to both feed-forward and cascade forward neural network for classification. The highest classification accuracy of 88.6% is achieved using 64 features and cascade forward neural network. Zheng et al. [2] develop

DOI: 10.24507/icicelb.12.12.1169

a method based on CNN to learn the features in the arc welding joint images and classify them. The total of 7460 images is randomly divided into training and testing sets according to the ratio of 80 : 20. Through the design of CNN architecture and optimization, the training parameters are optimized, and the method achieved an accuracy of 95.93% on test sets. Khumaidi et al. [3] use a CNN to predict welding defects using 10 layers of Gaussian kernel, 60 times iteration, and gradient descent algorithm. The Gaussian kernel, used for blurring images, improves the image extraction without losing the main information from the original image, and this filter also reduces the occurrence of noise or interference. This method has successfully predicted welding defects with validation accuracy until 95.83% for four different types of welding defects. Then, Zhang et al. [4] propose a Wasserstein Generative Adversarial Networks (WGANs) based data augmentation approach and train two deep CNNs using feature-extraction-based transfer learning techniques in imbalanced X-ray images of weld defects. The two trained CNNs are combined to classify defects through a multi-model ensemble approach, aiming to reduce the false detection rate. The experiments achieve satisfying accuracy, which substantiate the possibility that the proposed approach is promising for weld defect detection. Yang et al. [5] create a SqueezeNet-based CNN model to achieve more accurate recognition results compared with the state-of-the-art surface defect classifiers. Wen et al. [6] propose a deep CNN for semiconductor wafer surface defect inspection. This method uses feature pyramid networks with atrous convolution to extract semantic features and generate feature maps, and then feeds it into the Region Proposal Network (RPN) to generate region proposals. Then, the Deep Multi-Branches Neural Network (DMBNN) is used to classify and segment the defects. Kang et al. [7] use the novel CNN called Efficient Neural Architecture Search via Parameter Sharing (ENAS) to determine whether the architecture search method is effective for detecting the welding defect images.

Following the trend of the state-of-the-art for segmentation tasks and as inspired from the aforementioned research, in this study, we present a CNN model to segment weld spots. The architecture of the model is following UNet style [9] which is modified with various convolutional blocks and different sizes of feature maps. To be more detailed, the novelties of this research are summarized as follows.

- We create our own welding spot dataset that is gathered from several welding factories in Tulungagung city, Jawa Timur, Indonesia. Each factory has different welding characteristics which can be a challenging issue for any machine learning model. Each image is annotated using VGG image annotator [8]. This dataset can be used for future research in the area of welding segmentation.
- We perform an ablation study to evaluate the model performance by changing convolutional blocks on the model architecture. For each convolutional block, we compute the loss and segmentation IoU during training and validation to examine the best convolutional block.

The remainder of this paper is organized as follows. In Section 2, we describe the research methodology which includes the dataset information, model architecture, and experiment settings. Then, we provide the results and discussion in Section 3. Finally, the conclusion is presented in Section 4.

2. **Material and Method.**

This research aims to develop a deep learning model that modifies the UNet architecture to detect any weld spot by segmenting the captured image. In this section, detailed steps on how to develop the model are explained. In brief, this section describes how the dataset is created, the modified UNet architecture, metric and loss function, and the experiment setting.

2.1. **Dataset.**

A CNN model needs plenty of image data to be learned during the training process. As for the dataset, there are 34 videos of steel welding taken from several welding

(a) Image of weld spots (b) Annotated image (c) Generated mask

FIGURE 1. Weld spots dataset preparation

factories in Tulungagung city located in Jawa Timur, Indonesia. Each video is taken by a camera with a resolution of 1920 × 1080 (FHD) and has 30 Frames per Second (FPS). Then each video is extracted into several images with a range of 15 frames. This means that every 15 frames, a frame of video is saved as image data. In other words, there will be 2 images for each second in each video. With this mechanism, a total of 893 images are obtained. Several examples of welding images can be seen in Figure 1(a).

The next step of creating the dataset is to give an annotation for every weld spot on each image. In this research, VGG Image Annotator (VIA Tools) is used to draw the polygon to determine the region of weld spot on the image [8]. From 893 images, there are 12712 mask regions of weld spot. The annotated image with its weld spot region can be seen in Figure 1(b). Then, these annotations are converted into a segmentation mask programmatically using python3 programming language. These masks are used as the ground truth for training and validation. The generated mask can be seen in Figure 1(c). Black regions represent the background (class: 0) while the white regions represent the weld spot on the image frames (class: 1).

2.2. **CNN architecture.** In this research, UNet model is adopted with a lot of modifications to the architecture and the hyperparameter tuning in accordance with the needs of this research. In the original UNet paper [9], the architecture is divided into 2 paths which are the contractive path and the expanding path. The contractive path of the image is used to capture the discriminative features while the expanding path which is symmetric to the contractive path is used to provide precise localization.

However, in this research, the architecture is divided into 3 paths which are the contractive path, symmetric expanding path, and the bottleneck. The convolutional block is also different where each block looks like a rectangular shape instead of a square. This approach is made considering that the image size also has a rectangular shape. The modified UNet architecture can be seen in Figure 2.

Each rectangle box in Figure 2 is a tensor with dimension of H×W×C (height × width × channel) written in each box. The hollow arrow connecting between 2 tensors is a convolutional block with 2 × (3 × 3) convolution, batch normalization, and rectified linear unit (ReLU) activation followed by (2 × 2) max pooling (encoder side) or (2 × 2) billinear upsampling (decoder side) with stride (2,2) to adjust the height and the width of the feature map channels gradually. Meanwhile, the channel numbers of each convolutional layer are doubled so that the feature map channels are doubled on each block. In this architecture, batch normalization is applied to each convolutional block to make the training process become faster and more stable through normalization [10]. The dashed line is a skip connection to deliver the feature maps from the contractive path for concatenation in the symmetric expanding path. The filled arrow is also a convolutional block to produce the feature maps in the bottleneck. In this research, a comparison between several types of convolutional blocks for the bottleneck is studied. There are 5 different convolutional blocks compared in this research which are VGG block [11],

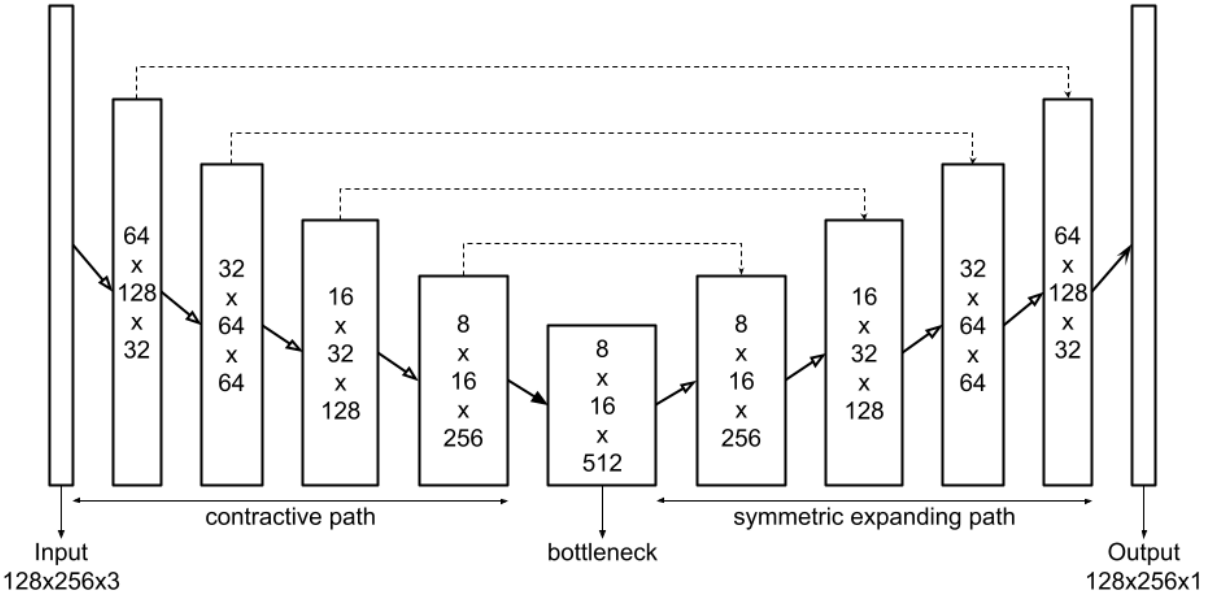


FIGURE 2. Modified UNet architecture

inception block [12], squeeze and excitation block [13], residual block [14], and dense block [15]. Finally, a point-wise (1×1) convolutional layer with sigmoid activation (notched arrow) is used to perform mask prediction. The number of output channels in the output layer is the same as the number of classes in the dataset which is only one (weld spot). The input size is resized to 128×256 with 3 channels representing the RGB of an image. Each convolutional layer in the contractive path and the symmetric expanding path has the kernel size of 3×3 with stride (1,1) and zero paddings are added to manage the size of the output feature maps for each layer.

2.3. **Metric and loss function.** In the segmentation task, a Binary Cross-Entropy (BCE) loss [16] combined with dice loss [17] is used as the loss function, and Jaccard Index or Intersect over Union (IoU) is used as the metric function. The equation of the loss function consisting of BCE and dice loss can be seen in (1).

$$\mathcal{L}_{BCEdice} = \alpha \left(-\frac{1}{N} \sum_{i=1}^{N} y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log(1 - \hat{y}_i) \right) + \beta \left(1 - \frac{2|\hat{y} \cap y|}{|\hat{y}| + |y|} \right) \quad (1)$$

where N is the total neurons in the output layer ($128 \times 256 \times 1 = 32768$) representing the width $\times$ height $\times$ number of classes, y_i is the pixel value in ground-truth y, and $\hat{y}_i$ is the pixel value in predicted output $\hat{y}$. We set both α and β to 1 so that each loss has the same influence on the model. In this research, a weight decay with $w_d = 0.0001$ is also used to penalize the model complexity and improve generalization. Thus, the total loss can be calculated with (2).

$$\mathcal{L}_{total} = \mathcal{L}_{BCEdice} + w_d \times \Sigma w^2 \quad (2)$$

where w is the model weights in the entire layers. The metric is also computed to justify the model performance during training and validation. We use the IoU score calculated with (3) as the monitored criterion to decide the learning rate reduction and early stop of the training process as described in Subsection 2.4.

$$\mathcal{J} = \frac{|\hat{y} \cap y|}{|\hat{y}| \cup |y|} \quad (3)$$

where $\mathcal{J}$ is the abbreviation for Jaccard Index or also known as the Intersect over Union (IoU), y is the ground-truth, and $\hat{y}$ is the predicted output.

2.4. **Experiment setup.** The training process is conducted in a GPU-powered virtual machine using cloud computing technique with the specification 1 × Tesla V100 provided by a Cloud Service. To obtain a model with better generalization performance, an augmentation technique is applied to pre-processing the image. The augmentation is a kind of rotation, resizing, normalization, saturation, contrast, and brightness [18]. For the training process, the batch size is set to 8 and each image is resized to 128 × 256 to fit into the model with 3 channels representing the RGB channels of the image. Thus, the batch dimension is 8 × 128 × 256 × 3. Then, the dataset is split into the training part and validation part with a ratio of 80 : 20. Thus, from a total of 893 images, there will be 714 images for training data and 179 images for validation data.

In the training phase, a batch of 8 images is fed forward to the model to obtain the predicted masks. These masks are compared to the respected ground truth to calculate the batch loss and batch IoU score. Then, both values are averaged and used to update the model weights during the backward propagation and early stop monitoring. This process is repeated until all batches have already computed. After calculating the average batch loss and average batch IoU, the average loss and average IoU over all batches in one epoch are calculated and recorded to monitor the model performance. Then, the process is looped over in a maximum of 1500 epochs and will be stopped if there is no improvement on the validation IoU within 150 epochs in a row. The average batch loss and the average loss in one epoch can be calculated with (4) and (5) respectively.

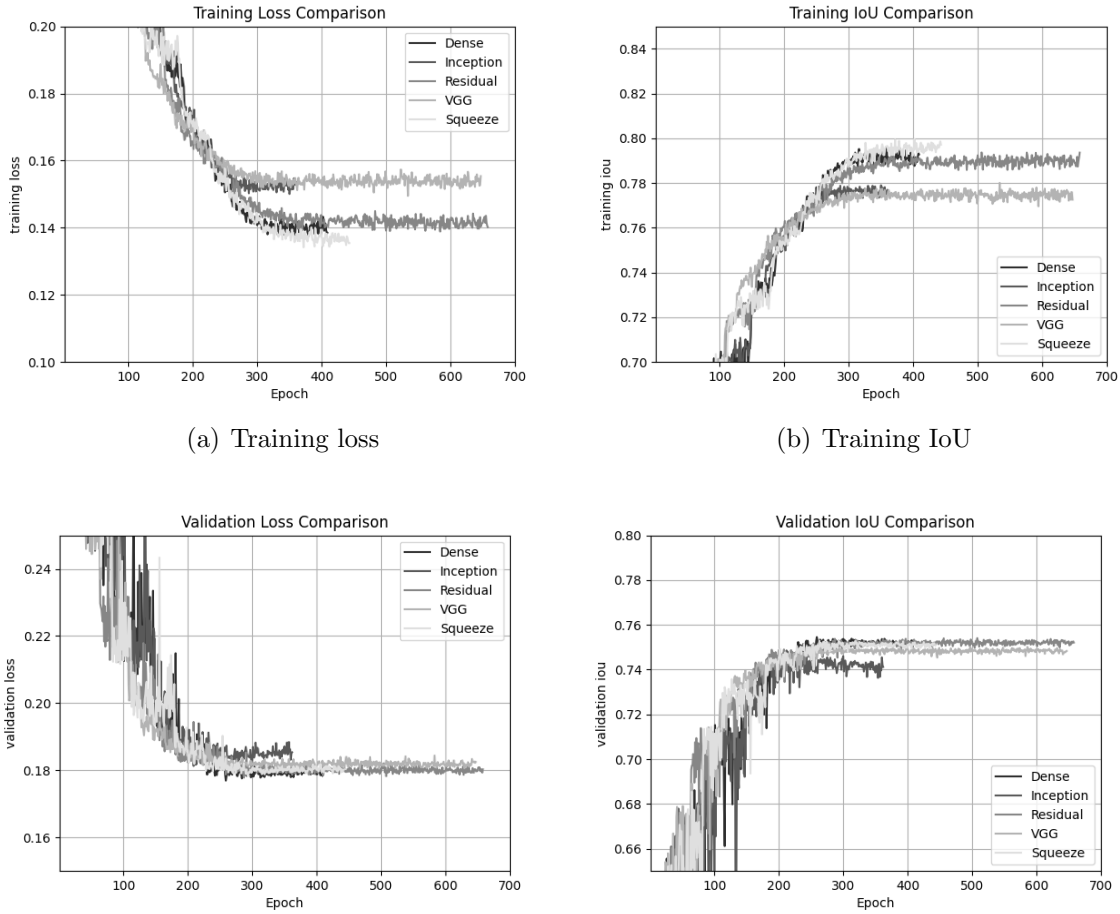
$$Loss_{batch} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{BCEdice} \tag{4}$$

$$Loss = \frac{1}{M} \sum_{i=1}^{M} Loss_{batch} \tag{5}$$

where N is the number of images in a batch (or the batch size) which is 8, and M is the number of batches. Meanwhile, the calculations for average batch IoU and average IoU are similar to (4) and (5). With a total of 893 images with the composition of 714 images as training data and 179 images as validation data, there will be 90 training batches and 23 validation batches.

Then, the overall process in the validation process is almost the same as the training process. The difference is that in the validation process, there is no backpropagation to update the model weights. As mentioned before, the average loss and average IoU on each epoch is used to determine when the learning rate must be reduced by half or when the training process will be stopped. Finally, a Stochastic Gradient Descent (SGD) algorithm used in [19] with the initial learning rate of 0.25 and a momentum of 0.9 is used to train the model until convergence. During the training process, the learning rate is reduced by half if there is no reduction in the validation loss in 15 epochs in a row. The maximum epoch is set to 1500 but the training process will stop automatically if there is no gain on validation IoU in 150 epochs in a row.

3. **Result and Discussion.** As mentioned in the previous section, the criterion that is used to determine the performance of the model is IoU and loss. During the training process, the model's performance on both training data and validation data is recorded. Therefore, there will be several logs of training loss, validation loss, training IoU, and validation IoU each for 5 kinds of convolutional blocks on the bottleneck. The comparison result of training and validation performance between 5 different bottlenecks on each criterion can be seen in Figure 3. It can be said that the model is not overfitting nor underfitting where both validation loss and validation IoU are close to the training loss and training IoU. As mentioned earlier, the training process will be stopped if there is no



(a) Training loss (b) Training IoU

(c) Validation loss (d) Validation IoU

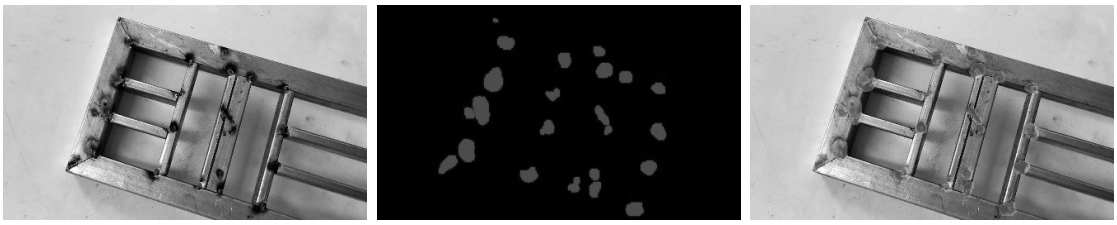
FIGURE 3. Training and validation results

TABLE 1. Performance summary

Conv. block	Highest training IoU	Lowest training loss	Highest validation IoU	Lowest validation loss	Total epochs
Dense	0.7960	0.1374	0.7545	0.1769	409
Inception	0.7789	0.1500	0.7485	0.1812	362
Residual	0.7936	0.1388	0.7540	0.1778	658
Squeeze	0.7996	0.1341	0.7530	0.1782	443
VGG	0.7800	0.1494	0.7502	0.1803	647

gain on validation IoU. This is the reason why each model can have different total epochs for the training process. For more details, the comparison result can be seen in Table 1.

The squeeze block has the best result on training data with an IoU score of 0.7996 and a loss of 0.1341 followed by the dense block as the runner up with an IoU score of 0.7960 and loss of 0.1374. On the validation data, the dense block has the best performance with an IoU score of 0.7545 and a loss of 0.1769 followed by the residual block as the runner up with an IoU score of 0.7540 and loss of 0.1778. Among all of the training processes, the inception block has the fastest convergence time with total epochs of 362 followed by the dense block as the runner-up with total epochs of 409. In the context of deep learning, model convergence means that the model has learned properly to respond to a set of training patterns with some error margin [20]. From those criterions, the highest IoU and lowest loss in the validation data are considered as the main criterions to determine the



(a) Input image (b) Predicted mask (c) Applied mask to frame

FIGURE 4. Inference result

best model performance. From the training process, it is known that the model's weights are updated based on the training data. Therefore, having the best result in training data but average or even bad performance on validation data means nothing. Having the fastest convergence time also means nothing if the performance of the model is not good enough since this criterion is ignored for deployment. However, if the model has the highest validation score on IoU or the lowest validation score on loss, it means that the model has better generalization when it is deployed. Thus, it can be concluded that UNet architecture with a dense convolutional block on its bottleneck has the best performance among the others since it has the best result on validation data. The other reason is that this model achieves top-2 on training IoU, validation loss, and convergence time. The model inference result on the test image can be seen in Figure 4.

4. **Conclusion and Future Work.** From the experiment, it is concluded that the UNet model with a dense convolutional block on its bottleneck has the best performance compared to the others. The dense block is considered as the best model based on the best result in validation IoU and validation loss which are chosen as the main parameters. Besides that, the dense block also achieves top-2 in the highest training IoU, lowest training loss, and fastest convergence time. The dense block has the highest validation IoU of 0.7545, lowest validation loss of 0.1769, runner up in training IoU, training loss, and convergence time with a score of 0.7960, 0.1374, and 409 epochs respectively. The reason validation IoU and validation loss are considered as the main parameter is because they indicate the model's generalization ability on unseen data (the data outside the training data) when the model is deployed. Meanwhile, training IoU, training loss, and convergence time are considered as supplementary parameters to judge the model performance since they indicate nothing for the models deployment and inference.

As for future works, there are several interesting things to be studied further. The first one is about extending the model's capability to recognize the welding quality and its defects. Formulating the algorithm for post-processing to count the number of weld spots is also an interesting study. Secondly, it could be interesting if all convolutional blocks on the entire architecture (in the contractive path, symmetric expanding path, and bottleneck) are modified or even using the Nested UNet (UNet++) instead of normal UNet as the main architecture. Then, the issue of hyperparameter tuning such as choosing the optimizer, learning rate, tensor dimension, and activation function, can be also challenging research in the future. Instead of tuning the parameter empirically, it would be good if the parameter can be tuned automatically or searched using a certain algorithm to find the best combination. Finally, when it comes to the deployment, the model size and complexity must be considered to achieve the fastest inference time on a specific device.

Acknowledgment. This work is supported by Direktorat Penelitian Universitas Gadjah Mada (grants number: 2403/UN1.P.III/DIT-LIT/PT/2020). The authors also acknowledge the helpful comments and suggestions of the reviewers, which have improved the paper.

REFERENCES

[1] J. Kumar, S. P. Srivastava, R. S. Anand, P. Arvind, S. Bhardwaj and A. Thakur, GLCM and ANN based approach for classification of radiographics weld images, *Proc. of the IEEE International Conference on Industrial and Information Systems*, pp.168-172, 2018.

[2] P. Zheng, J. Chen, S. Ye, P. Ott and L. Wang, Classification of arc welding joint images based on convolutional neural network, *Proc. of the IEEE International Conference on Anti-Counterfeiting, Security, and Identification*, pp.31-34, 2018.

[3] A. Khumaidi, E. M. Yunianto and M. H. Purnomo, Welding defect classification based on convolution neural network (CNN) and Gaussian kernel, *Proc. of the International Seminar on Intelligent Technology and Its Application*, pp.261-265, 2017.

[4] H. Zhang, Z. Chen, C. Zhang, J. Xi and X. Le, Weld defect detection based on deep learning method, *Proc. of the IEEE International Conference on Automation Science and Engineering*, pp.1574-1579, 2019.

[5] J. Yang, G. Fu, W. Zhu, Y. Cao and M. Y. Yang, A deep learning-based surface defect inspection system using multi-scale and channel-compressed features, *IEEE Transactions on Instrumentation and Measurement*, vol.69, no.10, pp.8032-8042, 2020.

[6] G. Wen, Z. Gao, Q. Cai, Y. Wang and S. Mei, A novel method based on deep convolutional neural networks for wafer semiconductor surface defect inspection, *IEEE Transactions on Instrumentation and Measurement*, vol.69, no.12, pp.9668-9680, 2020.

[7] M. G. Kang, H. Kim and D. J. Kang, Finding a high accuracy neural network for the welding defects classification using efficient neural architecture search via parameter, *Proc. of the International Conference on Control, Automation and Systems*, pp.402-405, 2018.

[8] D. Abhishek and Z. Andrew, The VIA annotation software for images, audio and video, *Proc. of the ACM International Conference on Multimedia*, pp.2276-2279, 2019.

[9] O. Ronneberg, P. Fischer and T. Brox, U-Net: Convolutional networks for biomedical segmentation, *Proc. of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp.234-241, 2015.

[10] I. Sergey and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, *Proc. of the International Conference on Machine Learning*, pp.448-456, 2015.

[11] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *Proc. of the International Conference on Learning Representations*, 2015.

[12] C. Szegedy, Going deeper with convolutions, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.1-9, 2015.

[13] F. N. Iandola, H. Song, W. M. Matthew, K. Ashraf, J. W. Dally and K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size, *Proc. of the International Conference on Learning Representations*, 2017.

[14] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.770-778, 2016.

[15] K. S. Sim and F. Sammani, Deep convolutional networks for magnification of DICOM brain images, *International Journal of Innovative Computing, Information and Control*, vol.15, no.2, pp.725-739, 2019.

[16] M. Shi, P. Dori and R. Reuven, The cross entropy method for classification, *Proc. of the International Conference on Machine Learning*, pp.561-568, 2005.

[17] C. Sudre, W. Li, T. Vercauteren, S. Ourselin and M. J. Cardoso, Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations, *Lecture Notes in Computer Science*, vol.10553, pp.240-248, 2017.

[18] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin and A. A. Kalinin, Albumentations: Fast and flexible image augmentations, *Information*, vol.11, no.2, p.125, 2020.

[19] D. Kreuter, H. Takahashi, Y. Omae, T. Akiduki and Z. Zhang, Classification of human gait acceleration data using convolutional neural network, *International Journal of Innovative Computing, Information and Control*, vol.16, no.2, pp.609-619, 2020.

[20] Z. Allen-Zhu, Y. Li and Z. Song, A convergence theory for deep learning via over-parameterization, *Proc. of the International Conference on Machine Learning*, pp.242-252, 2019.